Remotely Monitoring IIS Log Files

Rainer Gerhards & Dr. Tina Bird
rgerhards@adiscon.com, tbird@precision-guesswork.com
Adiscon GmbH, Mozartstrasse 21, 97950 Grossrinderfeld, Germany

Last modified 2003-03-07

Abstract

Web server log files are a very valuable source of forensic data for intrusion detection and overall network monitoring. Nevertheless, they are hard to integrate in a central system, especially in a real-time log monitoring infrastructure. In this document, we focus on Microsoft's Internet Information Server (IIS) log files and how to forward them to a central log repository. We describe the information contained in IIS server access logs, the way IIS log files are generated and a technique for forwarding them to a central log repository.

While this paper focuses on IIS log files, the same methodology can of course be applied to any other text log file on a Windows host, for example logs from Apache or DHCP.

Please note that this document is correct for IIS up to version 5.1. Microsoft has announced considerable change for IIS 6.0, and the information in this document might not be valid for that version.

Introduction

Web servers are the targets of a large and growing number of network-based attacks. Being widespread and known for a number of severe security weaknesses, Microsoft's IIS has become a primary target for malicious network activity.. IIS log files are an excellent source of information about probes and attacks, both for real-time system monitoring and for forensic incident analysis. Unfortunately, these files are stored in plain ANSI text files. IIS itself does not provide any way to forward them to any other system.

In this paper we present a summary of threats to IIS; a detailed analysis of the workflow through IIS, with emphasis on the generation of logging data; and a technique for collecting and reviewing those logs using a centralized logging infrastructure.

Threats

Vulnerabilities in IIS itself, badly written Web applications, and protocols such as the Simple Object Access Protocol (SOAP) provide a target-rich environment for an attacker. Both automated exploits (worms like Nimda and Code Red) and directed attacks can take advantage of any or all of these weaknesses.

Please keep in mind that these attack techniques are **not** limited to IIS targets, though they are the focus of this analysis.

Not all exploits can clearly be seen in IIS logs, but the majority of attacks seen in the wild leave signatures in the logs. Also, Web server log metadata – for instance, the number of incoming Web requests in a given period of time – can indicate problems or attacks. A sudden change in the number of connection requests can be a warning sign – maybe the only sign of a new kind of virus.

IIS Vulnerabilities

Most successful attacks against IIS occur because programming errors in IIS and its components allow an attacker to gain control over the machine or the web pages it hosts.

Please note that vulnerabilities occur in IIS filters, as well as in the core application. IIS has an extensible architecture. Filters from Microsoft and third-party developers provide additional functionality. Filters support technologies like Active Server Pages, as well as many others too numerous to mention. If you use third-party filters on your Web server, be sure to keep up to date on the vendor's security announcements – following Microsoft's security bulletins is **not** enough to stay secure.

It is also important to note that IIS has the typical processing of a) decoding a request, b) selecting the right virtual web server and c) directory and then d) calling the appropriate handler for the requested file. Then the web app takes over.

From the logging point of view it is important to note that IIS log files are written when the request is totally finished. This is the case because IIS logs include data like the number of bytes written to the client or the processing time of the request which is only available after the request is finised. As a result, a successful attack will not be logged if it is never finished.

As a side note, please be aware that IIS filters can potentially modify the log behaviour. The ISAPI interface used by filters provides mechanisms to modify the log file content as well as modify the location. Although I am not sure, I think a filter can even disable logging altogether. I haven't gone into much detail on these issues as I have not yet seen any wide-spread IIS filter that actually does one of those things.

In essence, this IIS workflow means there can be IIS attacks that will **not** be logged. The reason is that they occur before IIS even knows where to log. This is very seldom seen, but it definitely is possible.

SOAP (and .NET)

SOAP [1] – from the security administrator's point of view – is a protocol designed to circumvent firewalls. SOAP permits Web clients to make Remote Procedure Calls (RPC) into a distributed computing environment behind a Web server. Complex processing is easily encapsulated within a SOAP connection.

The key point here is that SOAP can easily be used to do malicious things, so it's definitely interesting to scrutinize inbound SOAP requests. From the IIS log perspective, we can start by checking for successful SOAP connection requests. More extensive research would require HTTP payload logging, which IIS does not do – at least not without some helper applications.

Please note that the "web services" introduced as part of Microsoft's .NET architecture are based on SOAP.

Badly Written Web Apps

Badly written Web applications generate all sorts of application vulnerabilities. These range from Microsoft FrontPage (for many people, still not an integral part of IIS) or the Microsoft Office Web extensions. But third-party software like *formmail* and ColdFusion also falls in this category.

We use *formmail* as an example to show that this problem is not limited to Microsoft products, despite popular perceptions. We currently see many requests to abuse a server with older versions of *formmail,* for spamming purposes. These attacks serve to emphasize the importance of monitoring

security announcements for all the software installed on a Web server, not just the code from Microsoft.

Also, please keep in mind that custom solutions – even if created by "expert programmers" – are frequently subject to well-known vulnerabilities.. A typical, common example is a query string parameter, representing an integer ID, that is passed to an SQL "where" clause without any validity check, exposing potentially-sensitive database content to unauthorized disclosure

IIS log files are very helpful in this scenario. You could parse, for example, for a "drop table" or other interesting stuff in the request parameters, to help identify this sort of attack.

IIS Log File Specifics

Name generation

Logfile name generation and management can be relatively flexibly configured. IIS supports hourly, daily, weekly and monthly log file rotation and naming. IIS also allows administrators to create a new log file based on a size threshold. Or if desired, the logfile can be left to grow without limit. In this case, a new log file is never created automatically, and the log needs to be rolled over manually or by an external process,a highly impractical approach in most production environments. .

In most cases, daily logfile rotation is adequate. Even on busy servers, logfiles typically do not outgrow a (dedicated) hard disk A day's worth of data is generally easy for both humans and processes to work with.

Please note that IIS is not similarly flexible when it comes to where the log files are stored. While a new base directory for all IIS logs is configurable, each virtual web server will have its own subdirectory beneath the configured log directory. Please note that the log directory is **not** the same as the web server's root directory[1]. The default system log directory is

"c:\winnt\system32\logfiles",

Web server logs are typically stored in the "W3SVC1", "W3SVC2", etc. directories. The numeric index is typically based on the order in which the web servers have been configured.

There is no known way to combine multiple virtual server logs into a single file.

How IIS writes Log Files

While IIS is running, the log file is kept open. Apparently for performance reasons, IIS does not write log files line by line but rather block by block. The log file is made up of 64 kb blocks.

When IIS appends the log file, it **always** appends a 64 kb block. The file area not yet containing log data is initialized with zeros. The actual lines of IIS data will later overwrite the lines containing zeros. IIS saves the log lines within the 64 kb block for an unknown length of time – apparently optimizing its performance by writing the data out when the system is idle. However, IIS seems to postpone the write at most until a block is full, so it is safe to assume that no more than 64 kb of log data will be written at a time. Also, please note that IIS has **never** postponed writing for more than some seconds during my testing. The actual log

[1] Storing the log directory beneath the web server's document root (that is, the root of the HTML files) is possible, but is definitely a bad idea from the security point of view. Such a configuration may permit an attacker to collect vast quantities of information about the requests, access patterns and user access to the server.

data inside the files can be considered as fairly close to real-time.

Therefore, while IIS is running, a file system query will always return a file size with a multiple of 64 kb. Since the logfile can be read up until and including the last block, a simple file read is likely to return meaningless data, the zeros padding the end of the file. More importantly, a process relying on the file growth to read and store data based on this growth will **not gather any log data at all**, as it will quickly read the (newly allocated) block of zeros. At the time when IIS rewrites the log block, the process will not see this as new data.

When IIS is stopped, the correct file size is adjusted in the file system. As such, whenever IIS shuts down, the file size will be reduced in almost all cases. This is normal behaviour and not an indication of an intruder trying to delete evidence. It may also confuse logfile rotation scripts.

Accessing Open Log files

Fortunately, IIS does not lock its logfiles while writing them. Therefore, a process trying to read the log file non-exclusively is able to retrieve the log file contents.

Contents

IIS supports several log file formats. The procedures in this document are valid no matter what log format is used. In the examples below we will be using the W3C Extended Log Format (http://www.w3.org/TR/WD-logfile.html), which is the IIS default.

Verify the properties and extended properties within your logging configuration. IIS uses the extended properties to define which data are to be included in the logs. In particular, if you plan on storing and monitoring your logfiles from a central server, configuring the extended properties to include the name of the server in its logs will make working with stored data easier.

HTTP Host Header Implications

Many web servers run multiple virtual servers on a single machine, or more precisely on a single IP address. A Web browser creating a connection request uses the HTTP 1.1 host header to specify which virtual host it is trying to reach.

This has an important implication for IIS exploits (like the well-known "Code Red" [4]). Most of the worms and other exploits circulating in the wild have propagated via IP address rather than URL or hostname. That is, they connect to the HTTP server **without** specifying a host header. This means that their connection requests are processed only by the virtual server that has **no** HTTP host header assigned. Most often, this is the default site, which should be removed by security-conscious IIS administrators. Does removing the default site mean that the server will not be vulnerable, since there is no virtual server configured? Unfortunately not! While most attacks will not work without a default Web site, a few do.

Given this situation, merely removing the default web site may no longer be the optimal plan. I now recommend configuring the default site, but removing all content, including – and most importantly – the mapped directories. This will enable you to gather logs and see almost all attack patterns, without jeopardizing the content on your Web site.

Similarly, more modern worms now tend to include HTTP host headers. This means that they will target virtual hosts, and are likely to leave signatures in the server access logs.

Requests with a non-matching HTTP host header are also processed by the default web site, so you will be able to see traffic from non-matching headers (if they are included) in your logs.

In addition, note that a number of exploits have leveraged IIS vulnerabilities in code areas before the virtual server selection. These exploits succeed without a particular virtual host being configured. If they managed to keep the IIS worker thread processing the request in a tight loop, **no log file had contained the malicious attack!**

Forwarding to the Central Server

IIS simply dumps the log files to the file system. There they sit until either the administrator reviews them or the intruder deletes them.

Especially in an environment with multiple IIS servers, consolidating the logs can be a hard task. And like most other system logfiles, IIS logs become much more valuable if you are able to correlate them with other network activity probes, like router and firewall data Even in a relatively small network, storing log files centrally provides benefits to security and system administrators.

We require IIS log files to be

- Forwarded to a central log host relatively quickly.
- Easily combined with other data on the central log host.
- Transported across the network without significantly increasing the load on the Web server.

As we have seen, we cannot handle this task with IIS features. We need to deploy third party tools.

Architecture

Agents

The requirements lead us to an architecture where an optimized agent program monitors the IIS log files on each (virtual) web server. Then, these agents forward the logs line by line to a central log server.

In large installations, depending on network bandwidth, number of virtual sites, and traffic load, a relay architecture involving several levels of loghosts (ie. branch office, corporate headquarters, NOC) may be the most effective logging infrastructure.

The key point for our purposes is that **each** web server will run a “forwarding agent”.

Alternatively, consider a central hub server running a “gathering agent”. Here, the hub would actively go out to the web servers, query and gather their log files remotely. At first, this approach looks tempting, since administration and updates to the logging system could be handled at a single system (the central hub). However, there are a number of drawbacks in this approach:

- There is a considerate amount of network traffic involved in polling the log files for new data. A centralized model requires all data to be transmitted across the network for processing. Additionally it loads the network with polling traffic even if a particular server has not generated new log data to be collected.
- The (central) agent relies on reliable network connectivity to the monitored server. If connectivity is lost, the agent is out of luck.

- The (central) agent cannot perform local actions on the monitored servers.

For these reasons we choose to implement a distributed architecture.

The decentralized agent model running on local web servers solves all of these issues. Obviously, network traffic is generated only when there are new log lines. But most importantly, the local agent can also reliably operate on a machine that is under attack. A local agent may also be able to protect local logs from tampering, and will preserve log data if network connectivity is disrupted. A local agent can generate a warning message to a different system log like the Windows Event Log. Local agents may also be combined with firewalls or other defensive technology to enable corrective action when an attack is detected.

Local agents require installing software on every machine to be monitored, but this installation can be automated relatively easily. The advantages of the decentralized architecture far outweigh the disadvantages.

In the rest of this paper, I will focus on MonitorWare Agent (MWAgent) [2].

Transport Protocols & Logging Infrastructure Issues

In a heterogeneous world, with log hosts running some flavour of Linux or UNIX, the obvious choice for a network-based log transport protocol is *syslog* (http://www.ietf.org/rfc/rfc3164.txt). *syslog* is designed to integrate log data from a variety of network devices into a central repository.

Unfortunately, *syslog* is UDP-based, and tends to perform badly on busy networks and servers. Servers get pretty busy when there is a worm attack.

There might be alternatives to syslog. For examplMWAgent can optionally use a non-standard protocol (SETP) which addresses many of the limitations of *syslog*. Obviously, this protocol cannot be used to talk to a *syslog*-based collecting system, and and is therefore only practical in a homogeneous MWAgent environment (at least inside a relay chain).

There is one limitation worth noting if RFC 3164 [3] *syslog* protocol is used. According to the RFC, *syslog* message size is limited to 1024 bytes. After subtracting the packet header and some overhead, that leaves roughly 980 bytes for actual log file data. Therefore, if IIS log files are transmitted by a fully compliant agent on a line-by-line basis, the agent needs to truncate any data after column 980. Typically, this should pose no problems, as URLs are not expected to become larger than around 200 to 300 bytes (and earlier versions of IIS core dumped anyhow if they were longer – without writing a log entry…). However, this standard limit should be kept in mind when gathering evidence for attacks with very long URLs.

In the rest of this paper, we assume the *syslog* transport protocol is being used, and that log data from remote sources is being collected and processed on a central log host. Building a loghost is outside the scope of this analysis, but is definitely required to make the most of the data being collected from a network. For more information, please consult the library section of the Loganalysis Web site, http://www.loganalysis.org.

Also note that in addition to the IIS access logs, the IIS host operating system logs (contained in the Windows Event Log) and logs from supporting systems (routers, firewalls, file servers, etc) are invaluable sources of information about potentially malicious behaviour. Recommendations for

audit configurations will be upcoming with the next release of this paper and are also available – for example - at http://www.monitorware.com/Common/en/Articles/Detecting-IIS-Intrusions.asp, http://www.sans.org/rr/win2000/harden_IIS.php, and http://www.lokboxsoftware.com/SecureWin2K/audit.asp (also emphasizes object auditing).

Monitoring free space on the Web server file system and network utilization to and from the Web server may also permit rapid discovery of malicious activity on IIS systems.

Local Warning Generation

We can use the local agents to generate alarms even before log data is transferred to the central server. This has the potential advantage of near-real-time notifications. It can also be used to save some additional evidence in case of a known attack, like adding important patterns to the Windows event log or forwarding them to another central system dedicated to tracking those attacks.

Performance Considerations

Web server logs can become quite large, especially on busy sites or in the face of automated attacks like Nimda and Code Red. Gathering and forwarding them can place "some" additional load on web server, the network infrastructure (like routers) and the log host. Unfortunately, there is no ultimate solution – or even guidance – for the performance issues of web log file monitoring. Therefore, the site administrator needs to evaluate the logging solution within his or her own environment. Monitoring web logs will have some impact on the server performance. It's **not** a more or less "zero impact" monitoring. This is especially the case for high-hit web sites, where monitoring needs to be very carefully planned.

Monitored Web Server

The performance impact of increased log monitoring on IIS servers derives from several sources:

1. Log files are periodically polled, and system resources are required to read the files even if no new lines have been added.

2. New messages must be formatted and processed for forwarding. This takes some CPU time.

3. Finally, the messages must be forwarded to the log host, which takes up network bandwidth.

Performance tests indicate that the polling and file reads have minimal impact on resource utilization. This result implies that using a short polling interval vs. a longer one is unlikely to create performance issues, at least on a dedicated web server. If a short interval is used, the high number of file reads places increased demands on the Web server. However, a longer polling interval, while reducing the amount of effort spent in overhead of file reads, is likely to greatly increase the amount of data to be processed and forwarded.

Depending on the load of your web server, it can be better to use a short polling interval (like 2 or 5 seconds). This causes a little more overhead, but ensures that server performance is used more or less as soon as there are new log entries created. Using a longer interval (e.g. 2 minutes) will result in an extended period (maybe some seconds), where the agent generates a high CPU utilization. During that period, the web server performance might be affected by the log forwarder.

The most important performance issue is the protocol being used to forward the messages once they have been formatted. We note that traditional UDP-based *syslog* generates many network packets and a noticeable load on the server while messages are generated.

This result is in part dependent on the transport protocol itself We have also tested Adiscon's proprietary logging protocol (SETP), which is TCP-based half-duplex communication (by intention). With SELP, the performance burden on the server is much better, as MWAgent is able to adjust to the actual receiver processing rate. This slows down the sender, providing time for other activities.

Most of to the performance degradation observed during testing of UDP *syslog* is due to UDP's tendency to send packets as quickly as possible. MWAgent (prior to 1.1SP1) does not specify a delay between log entries processed (as for example can be done with the event log reader). Versions later than 1.1SP1 have this capability. We recommend using version 1.1SP1 or later, and using a "overrun delay" of 1 millisecond to keep the *syslog* system impact low

Network Infrastructure

There is also another issue with sending the messages: no matter what the protocol is, you need to actually send the messages, which means creating network traffic. Fortunately, typical log file contents are far smaller than the HTML page being served for "this" log entry. Nevertheless, it **is** traffic and on a heavily used system, this might be an issue. For busy sites, we recommend installing an additional network card, dedicated to logging and other administrative tasks. Configuring the logging carefully also helps reduce system and network load: each unnecessary field in a line of log data creates unnecessary network traffic.

. Routers and other infrastructure devices differ in their ability to handle UDP traffic. On a busy Web server farm, setting up a dedicated network for logging traffic – perhaps using it for administrative and backup traffic if appropriate – can protect the "production" services from the negative impact of a spike in log-related traffic. This topology also helps shield potentially sensitve log data from attackers should part of the Web infrastructure be compromised.

Log Host

Assuming that you want to monitor IIS servers already in production, you can use your existing access logs to estimate the average amount of log data generated per day. The impact that new data will make on the log host depends on what the log host is doing with the data it receives – merely writing to a text file for archiving is much less demanding than evaluating complex rules in real-time. As a general rule of thumb, if the amount of Web server access data increases the total amount of data received and processed by your loghost by more than about 40%, you may want to consider upgrading your CPU and disk space, simplifying your processing (or splitting monitoring capabilities off of the system performing the archiving), or moving to a tiered log collection and analysis infrastructure..

Data rates for log hosts are frequently bursty, and this is nowhere more true than for Web servers, which are frequently subject to both malicious access (ie. Nimda) and inadvertent spikes of authorized traffic (the Slashdot effect). Like other logging clients, pre-version 1.1SP1 MWAgents generate bursts of traffic, which the loghost must record and process. The same is true for newer builds with "overrun protection" set to zero.

A scenario is always helpful. So here it is: we have a single IIS web server with a single

virtual server that should forward IIS log files to a *syslog* server. We will use MWAgent on the web server to gather the logs and transmit them via the *syslog* protocol to the log host.

Changes to the Loghost

No specific changes are required at the loghost, assuming that it has the capacity to handle the additional traffic introduced by the Web server. You may want to increase hardware capabilities if you have busy Web servers. You may also want to change your logging configuration to direct the Web server logs to other output files, or to specify particular actions be taken if Web attacks are detected.

Setting up the IIS Server

First, you need a copy of MWAgent [2]. A free trial copy is available for download at http://www.monitorware.com/en/Download/ Keep in mind that this paper is not marketing, but I simply need the product to do the trick.Once MWAgent is installed, you need to verify the IIS logging configuration, and set up MWAgent itself.

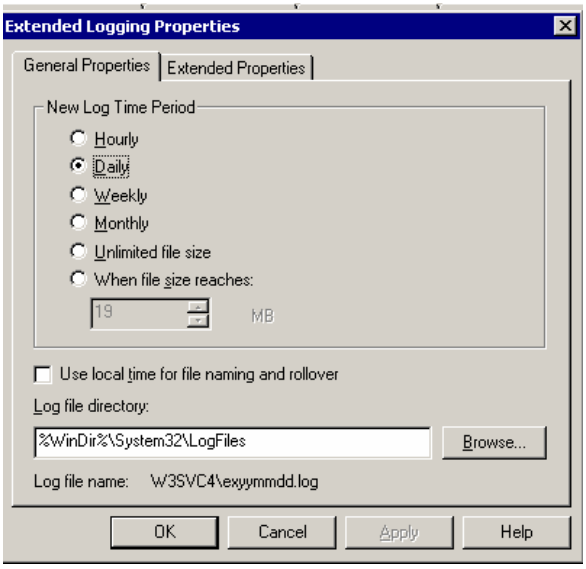
-

Configuring IIS

We need to configure IIS to

- write logs at all
- do this on a daily basis[2]
- include the necessary fields (host-name, for example)

The screenshot below shows the most important part of it, the daily log settings:



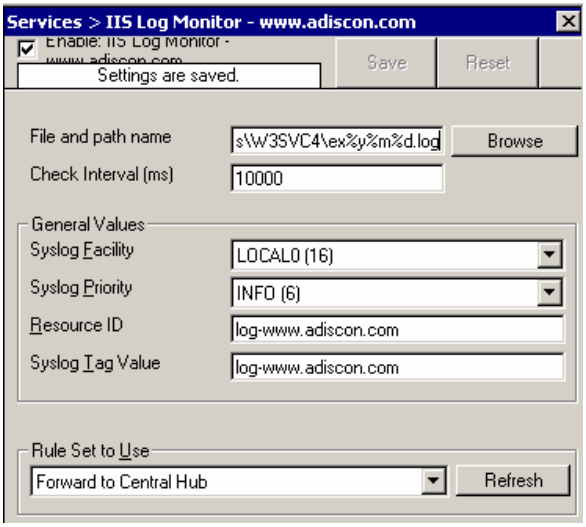
In addition to specifying how and when to create new log files, this field gives us the information to configure MWAgent, so that it is able to pick up the log files where IIS puts them. Please note the file name format. It is “exyymmdd.log” where “yy” is the two-digit year, “mm” the two-digit month and “dd” the two-digit day. IIS makes sure the numbers always have two digits in them, so January 1st 2003’s log file will have the following name: “ex030101.log”. Please also note that the file physically exists in the “W3SVC4” directory beneath the Windows LogFiles Directory.

Configuring MWAgent

MWAgent must now be instructed to monitor the log files IIS creates. Fortunately, MWAgent has been designed with IIS logs in mind and knows how to deal with them. Specifically, it addresses the way IIS writes log files (the 64 kb block issue) , and understands the IIS log file naming conventions.So all we need to do is point it at the log file and instruct it to monitor it.

[2] As I have written previously, we could use other formats, e.g. hourly. But our experience has shown that daily is quite effective and easy to handle, so we will use this in the sample.

This can be done with the "File Monitor" service in MWAgent[3].



Please have a look at the file and path name. The interesting part at the end reads as follows:

"`ex%y%m%d.log`".

Here we use placeholders: %y is the two-digit year, %m the two-digit month and %d the two-digit day. So on Jan, 1st 2003 the file name will be "ex030101.log" – exactly what IIS generates.

MWAgent handles the log file rollover at midnight (in this case). If that happens, the agent first checks if there have been any unprocessed entries in the old file before beginning to process the new file. This is done on a rollover-by-rollover basis. If IIS is stopped for two consecutive days (in this sample), the day in between is lost and not processed. Keep this in mind when planning your log files. This might be a good reason to use daily logs instead of hourly ones.

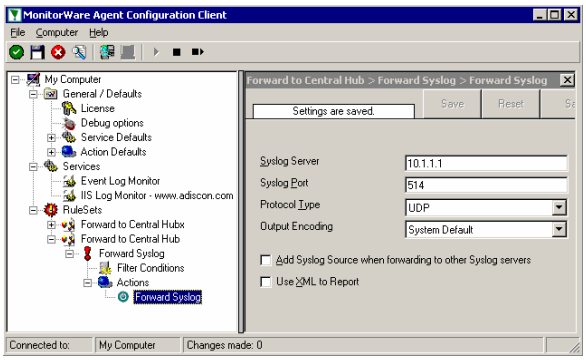
The "check interval" is expressed in milliseconds, so a check interval of 10,000 specifies that the log file will be checked for new entries every 10 seconds . The exact setting is a trade-off between performance used and degree of real-time monitoring required. For many servers, 5 to 60 seconds is not a bad value. MWAgent is optimized to require only minimal system resources, but you should not select a polling interval shorter than required for your environment. On the other hand, using check intervals longer than 1 minute is discouraged, because it leaves too much opportunity for an attacker to access a critical system without being detected. Also, in general, a shorter polling interval has its advantages: if the collector is run only once an hour (or even less frrequent), it may need to run for e.g. 5 minutes to process all data. During that period, the web server performance will probably be affected. In contrast, if it is run every 5 seconds, it will have a little more overhead (but only a little), but there will be no continous phase of processing and thus, all in all, less performance demand on the web server.

The general values should be pretty much self-explanatory, except for the "Resource ID", which is a MonitorWare specific value which we can ignore as long as we do not report to a MonitorWare Console..

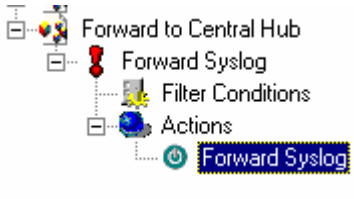
The "Rule Set to Use" field specifies a rule set for MWAgent, that carries out the actual actions on the events received (in this case, the log file line read). If an agent is not bound to a rule set, nothing will happen at all. For our example, we need to configure the rule set to forward log data to a central server.

So let us have a look at the respective rule set:

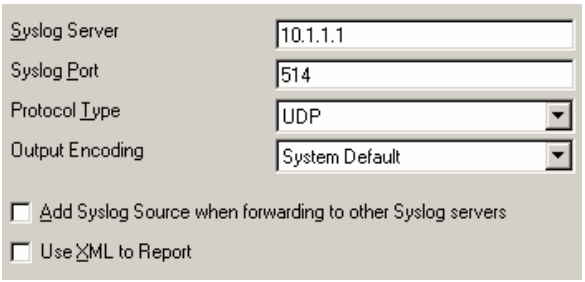
[3] I am not trying to duplicate the product manual here, as I think this is out of the scope of a general paper. If you need the specifics on services and MonitorWare concepts, I highly recommend having a look at the online help and manual. If you have never worked with MonitorWare Agent or WinSyslog 4.x+, be sure to read at least one of the step-by-step guides, it will definitely save you some time.



Obviously, this is a screenshot for those with excellent eyes. So let us zoom in for the rest of us. This is the relevant excerpt from the rule set:



Configuring the *syslog* server information is straightforward; note that MWAgent allows you to specify UDP or SETP for the transport mechanism, and to include the hostname or IP address of the system that generated the message in the first place:



The OutputEncoding is primarily for use with Japanese characters, so there is no need to change it from the default. The "Use XML to Report" option allow transmitting message meta data, but in this case, the 1024-byte *syslog* character limit really hurts. Therefore, we do not recommend using it in conjunction with IIS access logs.

Apply and save your changes to MWAgent, then restart it. IIS logs will immediately begin to drop into the central loghost.

Recommended Additional Magic

Once we're using MWAgent on the web server for forwarding access logs, we might also configure it to forward the Windows event log entries as well as disk space reports to the central *syslog* server.

For those interested in the capabilities of MWAgent and the possible additional "magic" that can be done, a brief overview can be found at

http://www.monitorware.com/en/Product/product_comparision.asp[4]

It typically takes less than a minute to open the browser window, look at the overview and close the browser again. Therefore, it might be worth a visit even for those who hate things that seem to be even remotely connected to marketing…

Once you've gotten your monitoring system in place you are in a position to detect attacks and problems with your IIS servers quickly. In an upcoming paper we will provide signatures of common IIS exploits, and provide information on how to detect them quickly.

Copyright

[4] Yes, there is a typo in the file name and yes, we believe in preserving URLs even if there is a typo in the file name ;-)

printout handed out). Please note that "unaltered" means as either the PDF file you (hopefully) acquired or a printout of the same on paper. Any other use requires previous written authorization by Adiscon GmbH and Rainer Gerhards.

If you place the document on a web site or otherwise distribute it to a broader audience, I would appreciate if you let me know. This serves two needs: Number one is I am able to notify you when there is an update available (that is no promise!) and number two is I am a creature of curiosity and simply interested in where the paper pops up.

Authors' Addresses

Rainer Gerhards
rgerhards@adiscon.com

Adiscon GmbH
Mozartstrasse 21
97950 Grossrinderfeld
Germany

Tina Bird
tbird@precision-guesswork.com

Stanford University
Polya Hall, Room 108
255 Panama Street
Stanford, CA 94305
USA

References

[1] http://www.w3.org/TR/SOAP/

[2] MonitorWare Agent – www.mwagent.com

[3] http://www.ietf.org/rfc/rfc3164.txt

[4] http://www.cert.org/advisories/CA-2001-19.html

Disclaimer

The information within this paper may change without notice. Use of this information constitutes acceptance for use in an AS IS condition. There are NO warranties with regard to this information. In no event shall the authors be liable for any damages whatsoever arising out of or in connection with the use or spread of this information. Any use of this information is at the user's own risk.