Global Information Assurance Certification Paper

Monitoring Web Server Logs Using Event Log Monitoring

By Steven Becker

GIAC Security Essentials Certification (GSEC)
Practical Assignment
Version 1.4b, option 2

Abstract:

In my current job the security department uses 'ELM Log Manager' by TNTSoftware to collect event logs and syslogs from a variety of servers and devices within various subnets around the network. In addition to collecting these event logs and syslogs, the ELM product has the ability to monitor text files on remote servers. This is done using the 'file-monitor- feature within ELM. In the initial deployment of this product at work use of the file monitor feature was out of scope and therefore not tested by anyone in our organization. After doing a little research and talking to others in the security department, it became obvious to me that the ELM file monitor is a feature that could provide the company with an additional layer of security and intrusion detection capability. When I approached management about this I was told that they agreed with me and that this was something that should be investigated. Unfortunately at that time there were no resources available at this to properly explore and test this in the lab. I decided that I would set up a lab at home where I could test the file-monitoring feature on my own. This would give the organization and myself a head start on testing when management decided it was time to pursue this. This would also provide a great learning opportunity for me and give me something to write about for this research project. This paper is a discuss and recap of what I did to prepare for the testing of this feature, the steps I went through during testing, as well as the results I obtained from my testing.

Before:

There are several advantages of a centralized log collection tool such as ELM. The main functional advantage is that it provides a single repository where a system administrator can easily check the logs of several servers. This is especially useful when the servers are in networks different then the network the system administrator is in such as a DMZ where other administration tools such as Microsoft Terminal Services may not be allowed to reach. ELM places an agent on each server being monitored and that agent makes a copy of every event log and delivers that log to the ELM server. This communication is encrypted and easily controlled since the service runs on predefined TCP ports, which can be easily monitored and controlled by firewalls and routers. Since the ELM agents make copies of log entries and send those copies almost immediately to the ELM server the ELM product has the ability to add to the

security audit scheme in a company. It is for this reason that my department chose to implement the ELM software. In the current configuration ELM collects the logs from several Windows servers and network devices. This information is then collected by the ELM server, which is installed in a security subnet, which very tightly controlled. By having copies of these logs the security engineers are able to do several things. First and foremost they are able to verify the integrity of the logs remaining on the servers by seeing what logs have changed or been removed. As we learned in my SANS Security Essentials course, generally the last thing a malicious user does after compromising a server is to clear their tracks by removing log entries. By keeping a secure and pristine copy of these logs the security team can see what has been altered. By having logs from several devices through out the network the security team can also use ELM to map out the "movements" of a malicious user as they moved from one device to another. This is very useful in determining the point of ingress for an attacker as well as what machines may have been tampered with. Since the information on the ELM server is very tightly controlled it can be used as evidence in criminal and civil court cases. One important thing to keep in mind when architecting a log collection and correlation scheme are system clocks. Generally log files contain the time they are created and this is the time stamp used to correlate them to log entries from other devices. The best way to make sure that all the systems have the same time on their system clocks is to point them all to a single time server. At my work we have installed an appliance timeserver that synchronizes its time directly from global positioning satellites (GPS). All the important servers and network devices are then synchronized directly from this timeserver. This guarantees that the log entries will all be stamped accurately and the entries when they get to the ELM server can be placed in their order of occurrence.

Many attempts to infiltrate a web server do not by default generate log entries since they do not immediately break anything or affect the operating system directly. For this reason the ELM file-monitor can be very useful. The file-monitor as I will test it, is set up to monitor web server log looking to specific patterns of activity usually associated with malicious attempts at access. Before I could begin to set up a lab I needed to decide exactly what I would do to test the file-monitoring feature of the ELM product. ELM ships with a sample file-monitor configured to monitor web server log files from Microsoft Internet Information Services (IIS) server, which is included in their Windows 2000 server software This made it fairly easy to determine that this is the type of server I should use as a "victim" to run my tests against. Since many of the servers at my work that are already sending event logs to the ELM server are IIS web servers, this would make the test results directly relevant to the current setup at work. Based on the results of my tests the IIS servers would be the likely candidates for using the file monitoring features. I learned in my SANS Security Essentials course that many attacks against IIS servers attempt to exploit buffer overflow so that arbitrary code will then be run by the server. I also learned that this code is frequently run as commands under the DOS command shell. With this in mind I made sure that

file monitor was actively checking for attempts to run "cmd.exe", the executable that launches the DOS command shell in Windows 2000, which is what my IIS server was running.

Having decided what my victim would be I then needed to decide what tests I would run against this server. I decided first that I would run four test scenarios, the first two against a default Windows 2000 Server installation with IIS and the other two with Windows and IIS fully patched with the most current patches recommended by Microsoft. The first test in each pair would be run without enabling the file monitoring features of ELM, while the second set in each pair would. This would allow me to create a baseline of what log info ELM captures when the software is configured as it is at work. Then any information caught by ELM with the file monitor feature turned on is obvious since it will be the difference between the logs captured in the second run and the first. In addition to testing the file-monitor feature this set of tests would allow me to see how patching the server affected log generation if at all.

One thing to note is that even a fully patched IIS server is not an exact replica of the web server at work. My initial installation of IIS matched what is installed at my work. Namely that only three of the IIS subcomponents are typically installed. They are the "World Wide Web Server", "Internet Information Services Snap-In", and "Common Files". This prevents unnecessary subcomponents from introducing new risks and potential vulnerabilities such as FrontPage Server Extension exploits or FTP brute force attacks. I did some research at work and found that most of the web servers have been hardened to one degree or another. I found that there is not a single standard for hardening, when a web server was deployed the hardening was done in what ever way the engineer installing the system thought it should be done. The level of auditing and logging is also left up to the engineers to determine at the time of install. As it turns out there is an effort going on between the web server administration group and the security group to create a set of standards for deploying web servers. These standards will capture the things that are already being done such as installing the minimum set of subcomponents needed to get the web server functional. These standards will also detail what security templates should be used and what the local security policies should be for the server. The part of these standards that will impact my work the most is the standardization of audit levels and logging levels for the servers. This will help to ensure that the correct information is being captured by ELM so that it can fulfill the security tasks it has been set up to do specifically collecting log and events that are related to malicious use of the servers. Because the auditing and hardening standards are still being developed and negotiated by the two teams at my work, I decided I wouldn't do any hardening but simply run the four tests I had already decided on. The results of my tests will help the security team know what information will be gathered by default and from that they can determine what may be missing and add that to the standard. This will also provide a baseline for later testing at work once the auditing and hardening standards have been finalized. By not

hardening my test server I run the risk that my tests the use live exploit code could crash the victim server before good data is gathered. Since this server is being run in a lab and not accessible from the Internet there is no risk that it could be attacked and altered by outside users.

To collect the data, I needed a server to run the ELM server and database. This server was a Windows 2000 server running Microsoft SQL Server 2000 and the ELM software. To generate log entries I needed to simulate attacks on the web server. To do this I opted to use the Nessus security scanner, which is a scanner that remotely scans a server looking for common vulnerabilities located in a database on the scanner. These vulnerabilities are grouped into common groups and are enabled by selecting a plug-in to test for the exploit. Each plug-in can be configured for more advanced users. I left all the plug-ins configured with their defaults for continuity. I also chose to use the whisker security scanner. Whisker is a PERL script CGI scanner developed by "rain forest puppy". The third tool I chose to use was HackIIS.exe. HackIIS.exe is a utility that simply tests for MSDAC vulnerabilities on IIS web servers. My main use for this tool was to provide me a way to quickly and easily get a hack attempt logged on the web server that contained "cmd.exe", one of the string the default ELM file-monitor searches for. These tools would give me an easy way to test most of the common exploits and vulnerabilities against the web server. To run these I used a Linux workstation and a Windows 2000 workstation.

During:

The first machine I tackled was the ELM server itself. Since the server at work has the ELM software and SQL software on the same server, that is how I installed it in my lab. The Windows 2000 Server and MS-SQL Server installations were all default to ensure operability. I then created a blank databse in SQL for ELM as specified in the ELM documentation. During the installation of the ELM server the ELM installation tool created the appropriate tables in the blank database and link it to the ELM software. Next I installed the ELM software as specified by TNTSoftware in their "Getting Started Guide". The ELM software has three pieces, the server, console, and agent. The server is the piece that collects the data from the servers and stores it in the SQL database. The console piece is a Microsoft Management Console (MMC) plug-in that is used to administer the ELM server. The agent is, as the name indicates, a small client piece that installs on the remote Windows servers to be monitored. On this server I installed the ELM server component, the console component, and the agent. This allowed me to administer the server as well as using the ELM agent to collect event logs from itself. To verify that the ELM software correctly installed I checked the SQL database. I verified that the installation created the correct tables in the ELM database. The installation created three tables one for alerts, another for knowledge base entries, and the third for the log entries

themselves. The last table hold the majority of the data produced by ELM and it is the tables that requires the most database administration and monitoring.

Once the ELM server was completed I moved on to the web server. This was a fairly simple installation of Windows 2000 server making sure I included the IIS components. I created a quick web page so I could verify from another machine that my installation was completed properly. I connected to the web server from a remote machine and verified everything was working by having the web browser on that machine load the page.

Next I set up the Linux workstation and Windows 2000 Professional workstation using standard default images. On the Linux machine I made sure Perl was installed and then installed Whisker and Nessus using the instructions included with each product. On the Windows workstation I installed HackIIS.exe and NessusWX the Win32 client for the Nessus server installation on the Linux box. I verified that the tools were working and ran each of them to make sure they didn't cause any systems to crash. Because of the invasive nature of some of the tools I was a bit concerned that they might cause the system to crash, especially if I enabled denial of service (DoS) plug-ins in Nessus. Even by enabling all the plug-ins on Nessus I was unable to crash the web server, which meant I could proceed with my testing and generate data without having to repair damaged systems. I also verified that these tools were in fact accessing the web server this was evident by manually looking in the web logs and looking for entries that they tools had connected to the web server. This was especially easy when looking at log entries created by Nessus since it identifies itself in many of the logs entries. I was also able to verify that the tools were in fact reaching the web server since they were able to find and report vulnerabilities.

Before beginning I formed a hypothesis as to what I expected to find. Since the ELM agent, as installed at my work, only collects event logs I didn't expect I would get many, if any, events generated when I ran my test tools unless these tools did something to affect the IIS service or the operating system itself. In order for ELM to detect "attacks" by these tools there would have to be Windows Events generated as a result. This doesn't happen by default, which is why the file monitor feature is valuable; it can create event logs based on finding key strings in the text file. I expected that after enabling the file monitor feature I would receive several events in the event log as it found "hits" in the IIS web logs. In my SANS course work we learned that patching a system can protect it from the majority of known vulnerabilities, but we didn't discuss in any detail how patching might affect the detection and logging of intrusion attempts. This was something I hoped to find out as an added bonus to the main objectives of the testing. Finally after all the systems preparations and forming my hypothesis I set out to runs the tests.

The first test was very straightforward. First I ran whisker and IISHack and verified entries were appearing in the IIS logs I then ran the Nessus scan with all 1080 plug-ins enabled (figure 1) which completed in just under 15 minutes.

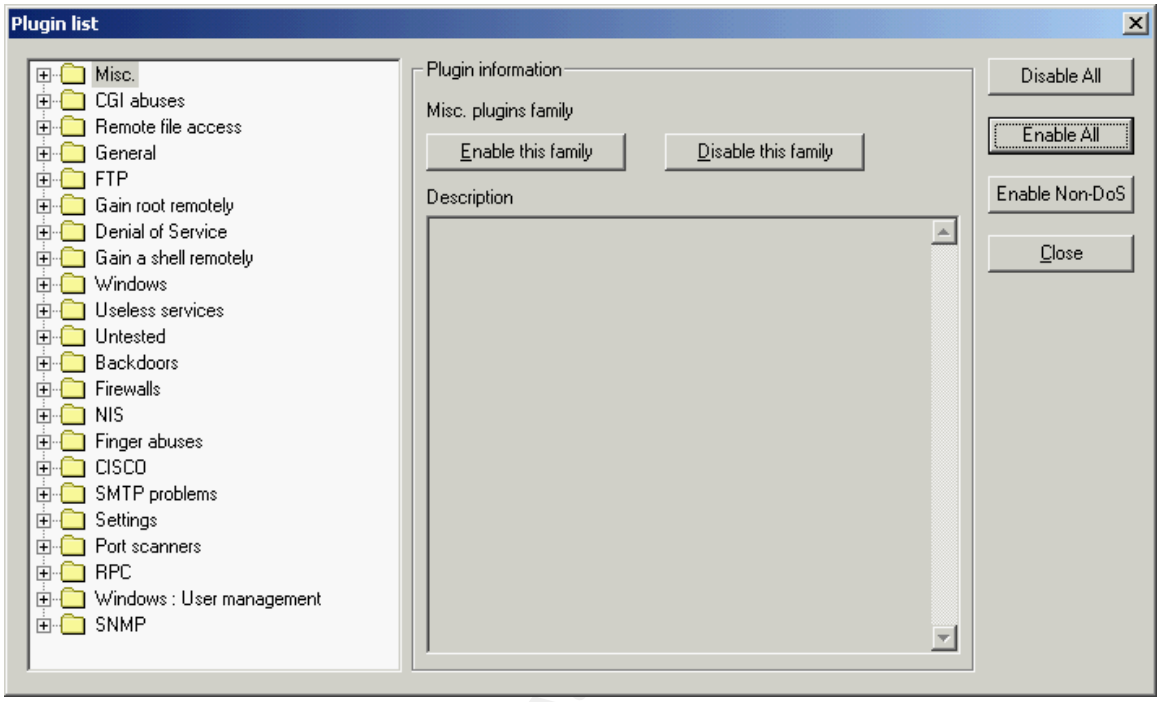

Figure 1

The initial Nessus scan reported that there were 55 total security holes with 6 of them being high severity, 36 low severity, and 13 informational. The most numerous security holes were a mix of Microsoft net-bios access being allowed and common www service exploits such as directory traversal and potential ISAPI filter exploits. Typical firewall implementations block all net-bios traffic from crossing into or out of the network in which the web server is installed so that these holes would effectively be false positives in a normal installation. Since my Nessus scanner was on the same network segment as the web server and not behind a firewall this traffic was not blocked and therefore flagged as a security risk. Some of the www security holes detected were also false positives since they require certain ISAPI filters to be running on the server. Since I did not install any ISAPI installed on the web server it should not be vulnerable to any of those exploits, I am not sure why the Nessus thought any of these were present on the server. It is nice to see that even a default installation of IIS does not produce an excessive number of security holes and that many of the thing that Nessus found can be controlled or eliminated by a simple firewall installation. Many times I have heard people complain that a default installation of IIS is "Swiss cheese" and full of security vulnerabilities. According to these test results it's not as bad as some have led me to believe. (This is of course not an excuse to avoid patching servers!) No entries were produced in ELM as a result of this test, which is what I had expected.

After saving the test results I applied all the available patches and updates available from the Microsoft Update site. For the second test I made a copy of the first Nessus session configuration to ensure that the settings were the same. I then confirmed that the web server was logging again using whisker and IISHack and checking the IIS log entries. After confirming everything was working I ran the Nessus scan again and recorded the results. This Nessus scan resulted in a total of 48 security holes being found with 3 high severity, 32 low severity, and 13 informational. This mix was pretty much the same as before patching the server except for their being 7 fewer security holes found. The three high severity security holes that were corrected by simply patching the server were a directory traversal exploit (CVE-2000-0884), an ISAPI buffer overflow exploit (CVE-2001-0508), and a script request decoding exploit (CVE-2001-0333). The four low severity holes that were fixed include two cross site scripting vulnerabilities (CAN-2002-0074), a directory enumeration vulnerability (CAN-2000-0071), and the fact that IP IDs were not randomized. Again as expected no ELM entries resulted from these tests.

Next I set up the file-monitoring feature in ELM. First I reinstalled Windows 2000 and IIS on my sacrificial web server so that it would be back to the original un-patched state. Then I reinstalled the ELM client this time selecting the file monitor piece as well as the event log piece (figure 2).

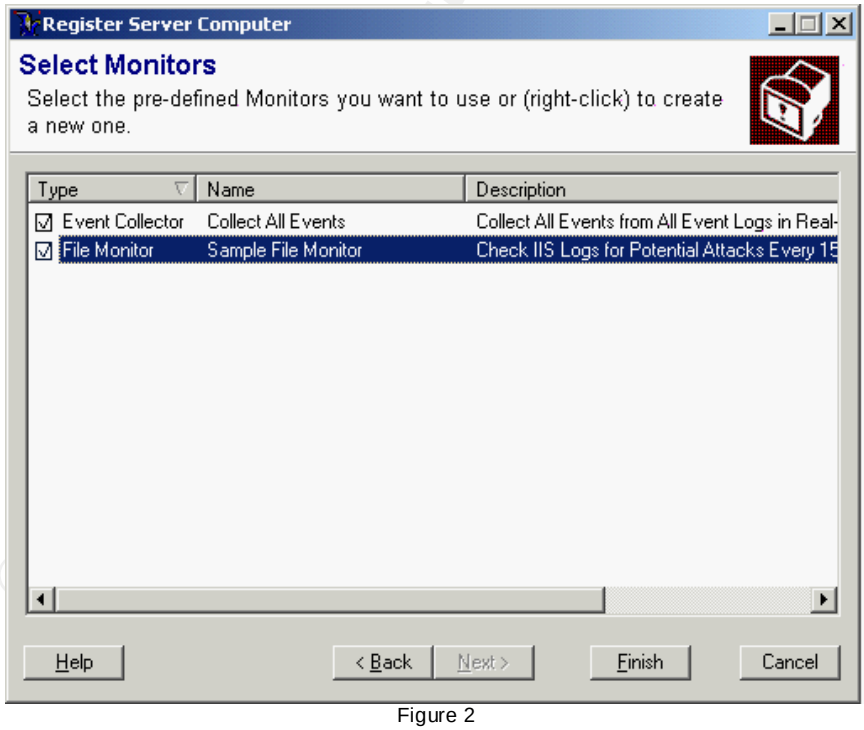

Figure 2

Next I enabled the monitor on the ELM server (figures 3 and 4) and ran the HackIIS tool to verify it all was working. This was verified by checking the web server log to see the entry where an attempt to run "cmd.exe" appeared and then

checking the ELM entries to verify that the file-monitor had created an entry based on the detection of the “cmd.exe” string in the log file.

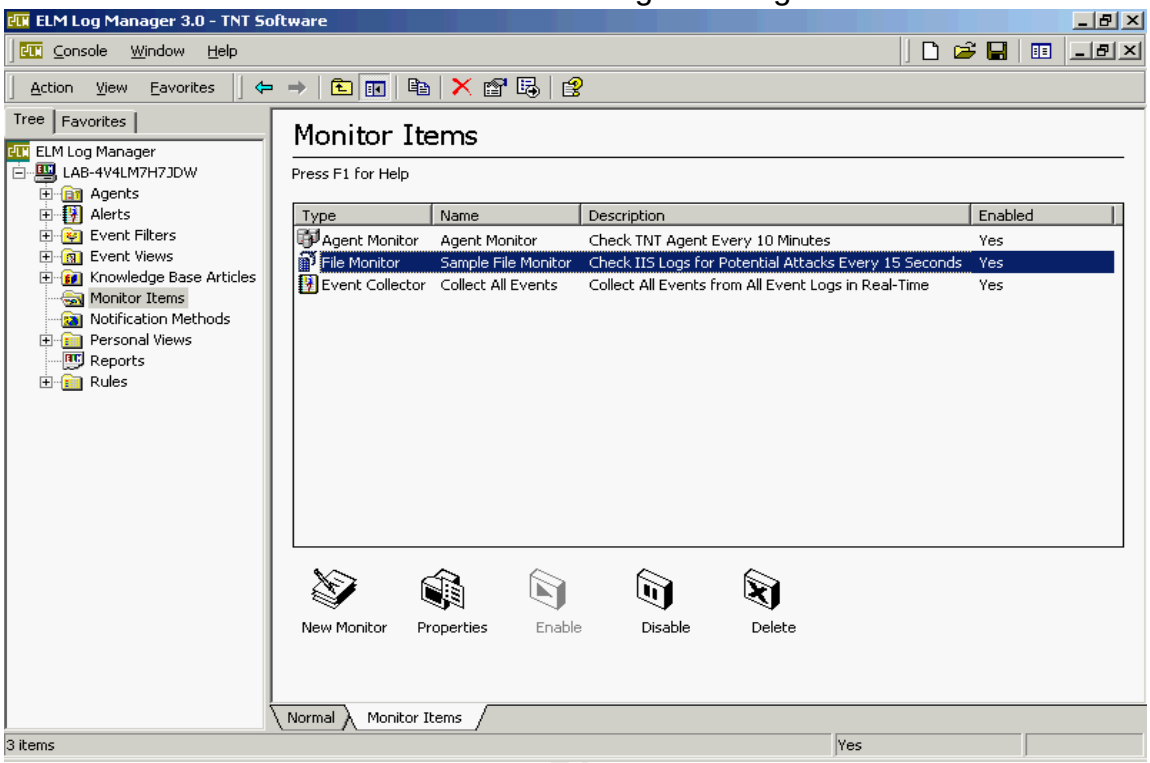

Figure 3

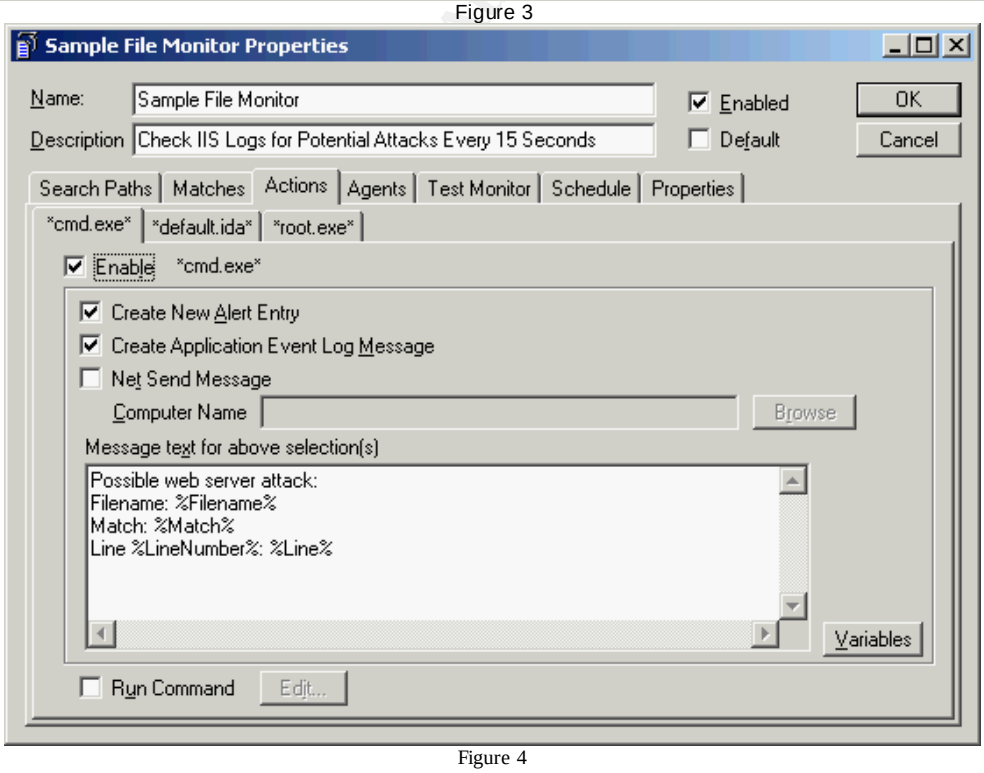

Figure 4

I had some initial trouble getting the file monitor piece on the server to sync up with the client piece. I contacted TNTSoftware and they helped me understand

how the file monitor creates a bookmark file that keeps track of what lines in the web log file have been checked and what have not. To help get the bookmark file synchronized with the web server log file TNTSoftware provided me with script in a batch file shown in table 1.

```
@echo off
setlocal
prompt %0$G$S

::FileMonHelper.cmd

FOR %%I IN (ex*.log) DO echo. >TEMP_%%I

echo EMPTY TEMP FILES CREATED
echo STARTING TO SLEEP FOR 15 SECONDS

sleep 15

echo DONE SLEEPING
echo APPENDING LOG FILES TO TEMP FILES

FOR %%I IN (ex*.log) DO type %%I >>TEMP_%%I

echo ALL DONE
```

Table 1

To use this script I had to also down load the sleep.exe utility from Microsoft. This script forced the file monitor book mark to sleep for a period time allowing new entries to be created in the log file and then appended a temporary file forcing the file monitor to see the new entries and cause the server and client pieces to resynchronize. To test that the batch file had synchronized everything and that the file monitor was reading the web server log I ran the HackIIS tool again. I verified that running the tool resulted in web server log entries and that a corresponding entry appeared in the ELM output, which it did. After confirming this was all working I began the second round of testing.

As I did in the first set of tests I ran Whisker and HackIIS to verify entries were being made in the web server log and in ELM, followed immediately by a Nessus scan. The Nessus scan settings were a copy of the original setting to make sure that all the test were selected and an exact duplicate of the earlier tests. Again the Nessus scan ran in about 15 minutes and the resulting reports indicated just over 40 security holes this time ELM reported that there were four suspicious web server log entries and generated a event log entry for each an example of which is in figure 5.

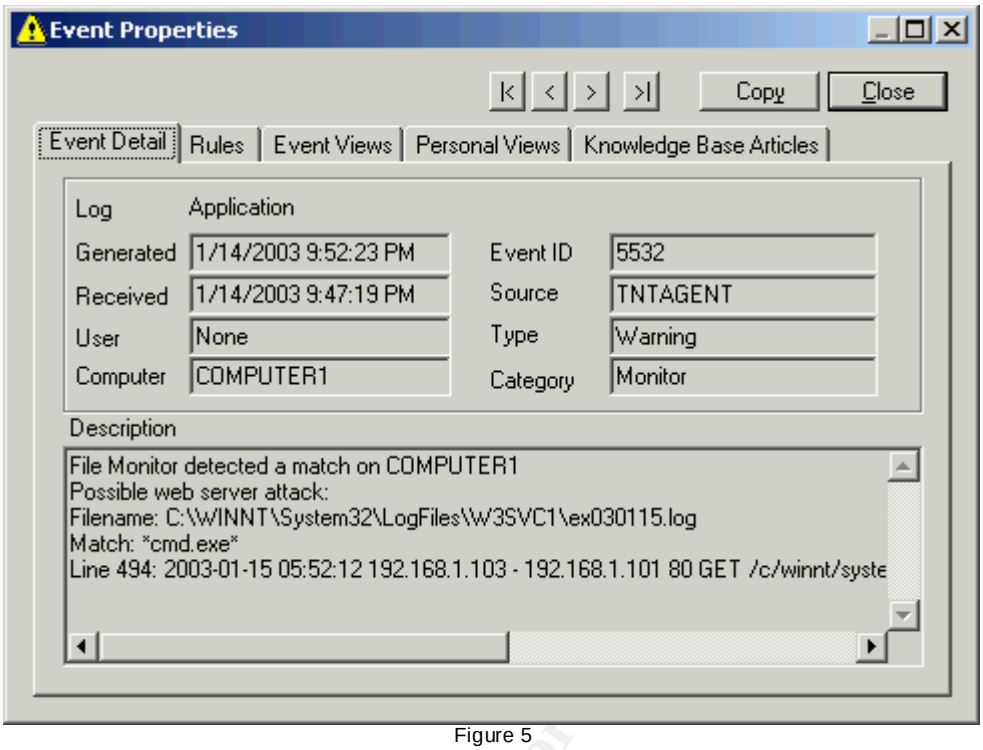

Figure 5

Next I applied the same Windows 2000 and IIS patches that I had before at prepared to run my tests again. I ran the batch file provided by TNTSoftware to synchronize everything after applying the patches tested it all by running Whisker and HackIIS and then ran the Nessus scan.

This time Nessus reported just under 40 security holes just as it did on the patched system before, but this time ELM generated over two hundred events. These events were all a results of entries in the web server log found to be suspicious by ELM. I was very surprised to see such a large difference in the amount of log entries created on a patched IIS web server versus an un-patched server. This large difference was such a surprise to me that I decided to run the second set of tests one more time to verify the accuracy of my results. This resting resulted in identical results confirming that patching the IIS server had a great impact on the amount of web log entries generating results in ELM. The table below is a summary of the test ran and the results in ELM (table 2).

Nessus Scan Results	Unpatched	Patched
Event Logs Only	ELM Entries: 0	ELM Entries: 0
Event and IIS Logs	ELM Entries: 4	ELM Entries: 202
Event and IIS Logs (repeat test)	ELM Entries: 4	ELM Entries: 202

Table 2

These increased number of ELM entries from the patched server were a direct result of increased web server log entries being generated on the server.

Something is Windows 2000 Services Pack 3 caused this increase in logging. There were 198 additional IIS log entries on the server when patched versus un-patched. Almost all of these additional entries were directory traversal attempts to run "cmd.exe". This explains why there was such an increase in the amount of ELM entries too since the file-monitor looks for each instance of "cmd.exe" being attempted. I tried to find more information on the various bug fixes and service enhancements in service pack 3 but was unable to easily identify which piece was responsible for the increase in logging. This is some thing I hope to do some further research on to get a better understanding of what exactly caused this.

After:

The results of the tests I ran were successful in helping me gather the information I hoped to gather. I was able to show that the file-monitoring feature of the ELM software worked and would be useful at my work to monitor IIS web server log files. I was able to show that by monitoring these log files we would be able to detect possible hack attempts at the host level. With this level of detection we could then use the alerting features in the ELM software to react to the hack attempt and stop the attack before it was successful or damage was done. Even if action is not taken during the intrusion attempt the ELM product with the file monitor feature will gather a substantial amount of log information that could be used to recreate the chain of events from a forensics standpoint and from an evidentiary standpoint provided we maintain the measures being taken by my work to protect the ELM data. This round of testing showed me that monitoring the event logs for key strings is a valuable tool in a "defense in depth" security strategy. The next step to further track whether or not the exploits were successful at running commands on the server would be to use a file integrity checker discussed in my Security Essentials course, such as Tripwire. This would not only show if a hacker had been successful but it would also show conclusively what if anything had been accessed and changed by the hacker. File integrity checkers such as Tripwire do this by maintaining a database of all the files, directories, and registry keys being monitored on a server. This database contains information on the size of the files, directories, and keys as well as information such as the last time they were accessed, deleted, etc. This information is kept encrypted and serves as a very definitive way of determining whether or not a specific file has been altered.

The next step for testing the features ELM has and how we can use them at work is to test the alerting features in the ELM product. Since I have shown that ELM will detect the intrusion attempts the next logical step for the product is to alert us to the attack in real time. The ELM product can also react automatically based on the events collected. This feature would allow us to not only detect and alert based on a particular event but also to take automated steps to stop the attacker by running a command script that could be used to kill the connection to

the offending IP address. I have done some preliminary testing by creating some simple alerts that send SNMP traps or SMTP messages. Either of these could be used to send a real time notification to a security engineer to alert them that a server was being attacked. ELM also has the ability to run user defined scripts or program in response to certain traffic. This could be used to increase the amount of logging for a certain IP address when malicious activity from that address is detected. This feature is one I have not yet had a chance to test but plan on testing very soon.

In addition to helping me gather the data I hoped to gather and answering the questions I hoped to gather, this project also raised some new interesting questions. The first question raised is why patching the IIS server did not have as great an affect on the Nessus scans as I originally thought it might. Part of that is that the default installation wasn't as vulnerable as I first suspected. Many of the other vulnerabilities detected by Nessus were services such as net-bios that are easily blocked by firewalls or even disabled on the server itself if not needed. It would also be interesting to see how the Nessus scan results changed after hardening the IIS server using the knowledge I gain in my SANS training and in accordance to the SANS publication "Securing Windows 2000: Step by Step". The next question raised by this testing is why the patched IIS server created so many more log entries then the un-patched server did. Further testing and investigation would be interesting to see what piece or pieces of Windows 2000 Service Pack 3 increased the amount of logging in IIS. These questions were partially answered during this project, but will require project of their own at a later time for me to fully investigate them.

References:

Apolonio, Larry. "Downloads." Welcome www.ipchains.net. February 8, 2001. URL:http://www.ipchains.net/Downloads/downloads.html (1 Jan. 2003).

Deraison, Renaud. "Nessus." The Nessus Project. February 24, 2003. URL:http://www.nessus.org (26 Feb. 2003).

Microsoft, Inc. "Microsoft Windows Update." 2002. URL:http://windowsupdate.microsoft.com (12 Jan. 2003).

Microsoft, Inc. "Microsoft Windows 2000." Windows 2000 Server: The multipurpose network operating system for small businesses and workgroups. February 19, 2003. http://www.microsoft.com/windows2000/server/default.asp (23 Feb. 2003).

Rain Forest Puppy. "Whisker information, scripts, and updates." December 18, 2002. URL:http://www.wiretrip.net/rfp/p/doc.asp/i2/d21.htm (1 Jan. 2003).

Reinstein, Robert. Practical Microsoft Windows 2000 Server. Indianapolis: Que, February 2000

Scambray, Joel and McClure, Stuart. Hacking Exposed Windows 2000. McGraw-Hill Osborne Media, August 29, 2001

TNTSoftware. "ELM Log Manager™ 3.0." ELM Log Manager™ 3.0: How the first-to-know stay ahead™.
URL:http://www.tntsoftware.com/products/ELM3/ELM30/default.asp (12 Jan. 2003).

TNTSoftware. "ELM Log Manager: Getting Started".
URL:http://www.tntsoftware.com/products/ELM3/ELM30/ELMGettingStarted.pdf (12 Jan. 2003).