Digital Forensics Professional

LOG ANALYSIS

Module 8

v1.0

Log Analysis Module – Map

MAP REF

8.1 Introduction

8.2 Logging Infrastructure

8.3 Using Linux Tools for Log Analysis

8.4 Web Logs

8.5 Windows Events

8.6 Syslog

Introduction

8.1 Introduction

If there was an “Investigator’s best friend” award, it would probably go to log files and log analysis tools.

For an investigator with the right knowledge, log files are books telling the whole system’s story.

A log is a file that is used to document all the events, actions and/or errors that an application or device encountered.

Logs were originally used as a way to keep track of the errors a system produces and troubleshoot them.

Later, it became clear that those files have great forensics value and can be used during investigation to rebuild the crime actions and timeline.

Unfortunately, logs are often under-appreciated, even though they are a very useful source of information:

- Computer system resource management:
  - Printers
  - disk systems
  - battery backup systems
  - operating systems, etc
- User and application management:
  - login and logout
  - application access, etc
- Security

Sometimes the type of information can be categorized into more than one bucket

- User login and logout messages (management and security)

Various disk storage products will log messages when hardware errors occur. With such information we can resolve small problems before they become really big nightmares!

What is Log Data?

- The heart of log data are “log messages” or “logs”.
- A log message is what a computer system, device, software, etc. generates in response to some sort of stimuli.

A stimuli depends on the source of the log message.

Examples:

- Unix systems will have user login and logout messages
- Firewalls will have ACL accept and deny messages
- Disk storage systems will generate log messages when failures occur

Logs information can be categorized into the following groups:

- Informational
- Debug
- Warning
- Error
- Alert

8.1 Introduction

Logs are crucial in almost any investigation as they provide the investigator with insight on the targeted system events, actions, connections and errors.

Logs come in different formats and structures like simple text files such as **Apache** and **IIS** logs, which can be examined using text editors, to preparatory logs such as Windows Event logs, where you need special utilities to read and examine its content.

Log analysis (or *system and network log analysis*) is an art and science seeking to make sense out of computer-generated records (also called log or audit trail records). The process of creating such records is called data logging [1].

In order to perform log analysis, one must know where the targeted system stores its logs in addition to knowing the format and structure of the log file and log data.

Such knowledge is essential and would serve the investigator in reading the log and writing a parser that parses the log.

After a log is generated, the next step is to filter and normalize the messages

Filtering deals with including or excluding log messages based on the content in the log message

- Deciding what to filter greatly depends on your organization's needs

Normalization is the act of taking disparately formatted log messages and converting them to a common format

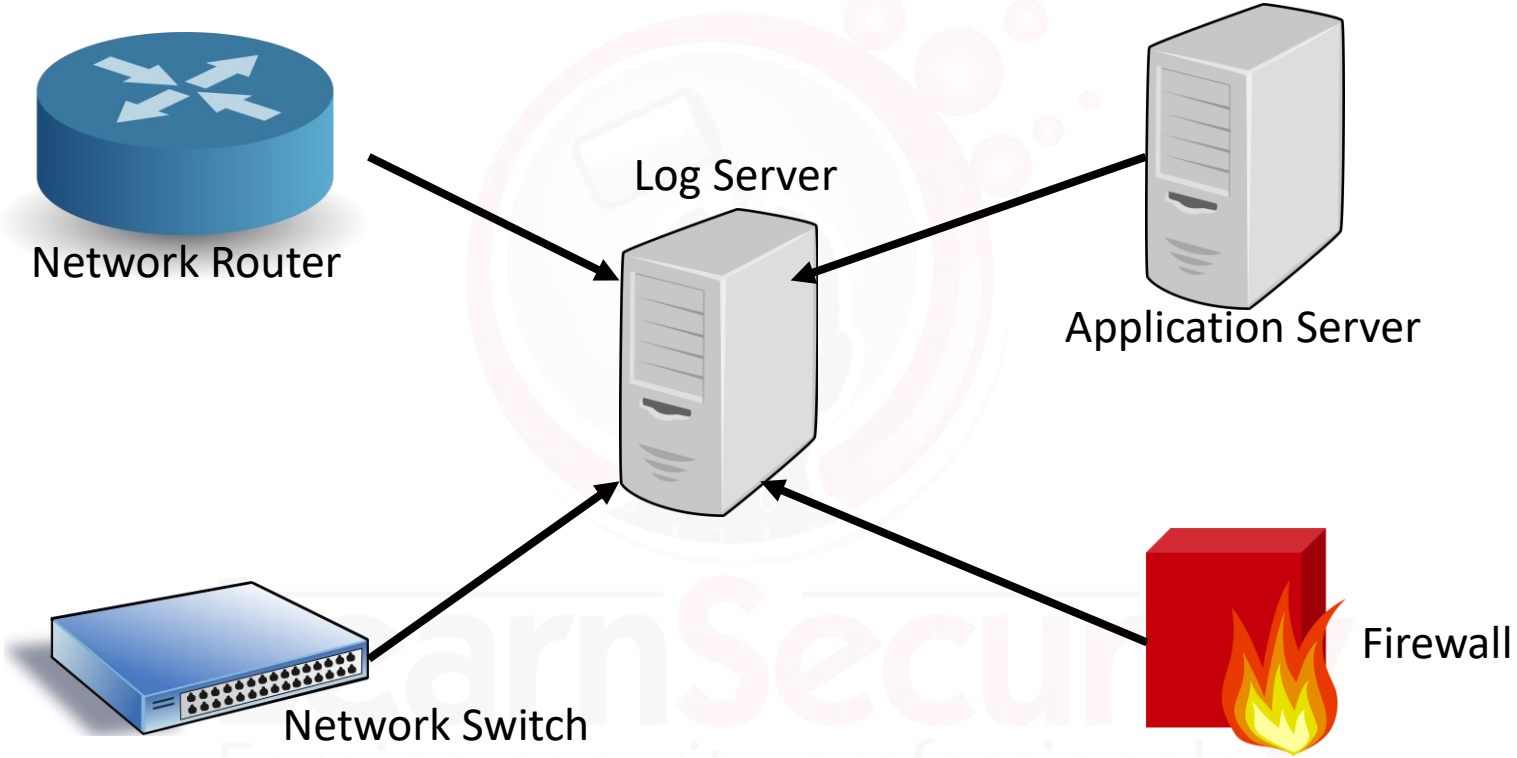
Logging Infrastructure

8.2 Logging Infrastructure

MAP REF

The design of the logging system depends on the size and the structure of the network.

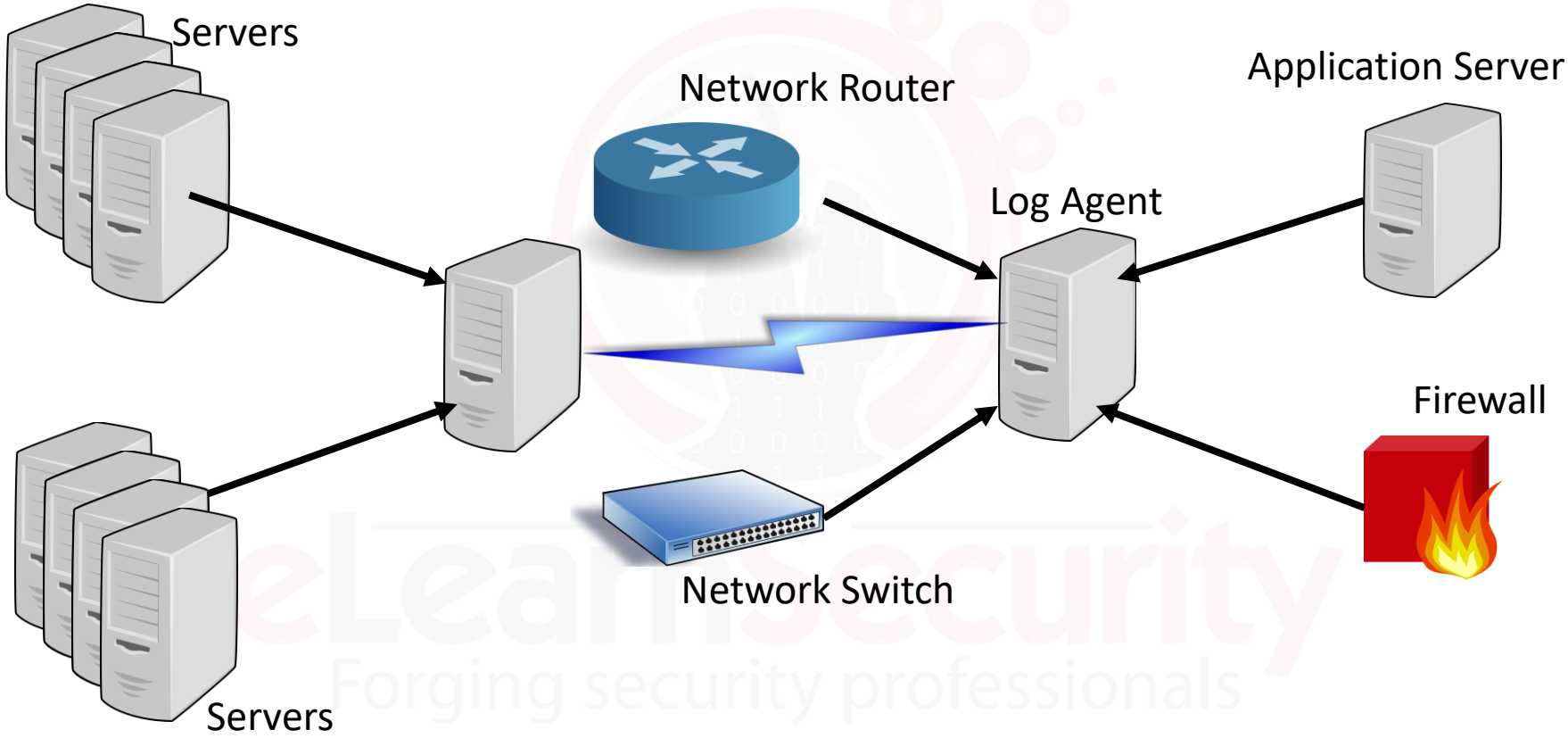
For example, in small networks, a basic logging mechanism relies on a logging server which acts as the center of logging system where all the networking devices send their logs to the server for storage and analysis.



The previous model is simple and easy to build. However, it doesn't scale for a large enterprise network that is usually made up of many small size networks.

In that case, each network segment has its own log aggregator (also called collector) which collects the logs from the devices on the network segment and sends it to a centralized logging server.

8.2 Logging Infrastructure



Using Linux Tools for Log Analysis

8.3 Using Linux Tools for Log Analysis

Luckily for us, Linux provides us with many built-in tools to read and format the output of log files and text files in general.

This section will cover few of the most used commands to read and format the output of text files.

8.3 Using Linux Tools for Log Analysis

Without the need for installing additional tools, we can take advantage of Linux's built-in utilities and **command pipelining** to view and extract data from logs.

8.3 Using Linux Tools for Log Analysis

The **cat** command displays the content of a text file on the terminal's screen.

```
root@kali:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

However, the cat command alone won't always be enough on its own, as the output sometimes is too messy for us in case we're looking for a certain line.

To extract a certain line from the output we can use the **grep** command along side the cat command.

In order to use the **cat** and the **grep** command together, we need to pipeline the output.

Pipelining in Linux means making the output of the first command as an input for the second command.

Instead of getting the full output from cat, we can redirect it to **grep** and ask it to display only the lines which contain a certain word.

8.3 Using Linux Tools for Log Analysis

To understand how pipelining work, let’s take an example on pipelining the output of the famous **NMAP** network scanner.

We’re going to try and perform network sweep on the local Lan we’re at to know how many live machines are there on out network.

8.3 Using Linux Tools for Log Analysis


```
root@kali:~# cat scanresult.txt
Starting Nmap 7.01 ( https://nmap.org ) at 2017-10-20 00:51 EDT
Nmap scan report for 192.168.153.1
Host is up (0.00059s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.153.2
Host is up (0.00019s latency).
MAC Address: 00:50:56:EB:F7:91 (VMware)
Nmap scan report for 192.168.153.254
Host is up (0.00043s latency).
MAC Address: 00:50:56:F4:DE:34 (VMware)
Nmap scan report for 192.168.153.133
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 2.38 seconds
```

We can see that the output is little messy. What if, for some reason, we want to output only the IP addresses instead of the MAC address and the rest of the output?

What if we have a tool that will read NMAP's output, and it expects to read IP addresses only? This is where **grep** and **cut** will come in handy.

Let’s examine the output of **NMAP** to know how we can handle it. NMAP displays three lines for each live host. The first line states which IP it is scanning, the latency of the host’s response and the mac address.

We need to extract the lines which contains the IP address only.

We'll type the command again. However, this time we will redirect the NMAP's output to be the input of the grep command.

In Linux, the | is used to pipeline.

The command will be :

Cat scanresult.txt | grep “report”

```
root@kali:~# cat scanresult.txt | grep report
Nmap scan report for 192.168.153.1
Nmap scan report for 192.168.153.2
Nmap scan report for 192.168.153.254
Nmap scan report for 192.168.153.133
```

We chose to grep the line which contains the “report” word.

The output of that command will filter out most of the unwanted output we saw earlier.

However, we still do not have what we originally wanted. The output still gave us unwanted text which is the “Nmap scan report for” phrase.

We managed to **grep** the phrases we wanted from the output, now we need to **cut** the output into parts (we'll call them fields) and extract the IP address field only.

To do so, we can use the **cut** command which takes a delimiter to cut the phrase using it.

A delimiter is a separator between the phrases field.

For example, in the IP address, the dot "." serves as a separator between the octets of the address.

In normal phrases, the space " " is what separates the words of a sentence from each other.

We can specify the delimiter for the cut command using the -d option.

cat scanresult.txt |grep report| cut -d " "

After specifying the space as a delimiter, the phrase will be separated into fields based on the delimiter we used.

Field#1	Field#2	Field#3	Field#4	Field#5
Nmap	Scan	Report	For	192.168.153.1

The final step would be to pick the field we want from the phrase.

This is done using the **-f** option and the number of the field we want to display.

Since we want to display the IP address only, that means we need to extract the 5th field.

```
root@kali:~# cat scanresult.txt | grep report | cut -d " "  -f 5
192.168.153.1
192.168.153.2
192.168.153.254
192.168.153.133
```

A more comprehensive tool that we can use to control the format of the output of a file is awk.

AWK is a programming language designed for text processing and typically used as a data extraction and reporting tool. It is a standard feature of most Unix-like operating systems [2].

We can apply talk as a command directly like any other command on Linux, or we can write it on a script and run it to extract data for us.

Much like cut, when opening a file we need to determine a separator which we will rely on to separate the fields in our output.

For example, if we want to write an **awk** command to display the users on a Linux system, we can open the **/etc/passwd** file and user the ":" as the separator between our fields since the name and the other fields on each line is separated by : on the passwd file.

We'll specify the separator using the -**F** option, and write the format which we want to display the output in between single quotes.

8.3 Using Linux Tools for Log Analysis

In this command, we want to print the firsts field (denoted by $1). We can print any fields we want and separate them using the (,)

```
 GNU nano 2.8.7                File: script.awk
awk -F ":" ' { print $1 } ' /etc/passwd
```

8.3 Using Linux Tools for Log Analysis

MAP REF

Here are the results.

```
root@kali:~# ./script.awk
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy
www-data
backup
list
irc
gnats
nobody
systemd-timesync
```

We can also specify the records we want to display using the **NR==** option.

For example, if we want the records between the second and the 10th record we would add NR==2,NR==10 before the printing statement.

8.3 Using Linux Tools for Log Analysis

In addition to displaying and manipulating the format of an output, AWK contains many built-in functions we may use to help us format our output.

For example if we want to display the length of each username alongside each user, we can use the print function and add it to our command.

```
  GNU nano 2.8.7                File: script.awk
awk -F ":" 'NR==2,NR==10 { print $1,length($1) } ' /etc/passwd
```

8.3 Using Linux Tools for Log Analysis

Here are the results.

```
root@kali:~# ./script.awk
daemon 6
bin 3
sys 3
sync 4
games 5
man 3
lp 2
mail 4
news 4
```

If you're familiar with the C programming language you'll find the AWK formatted string familiar.

The **printf** function, allows us to specify the format which we want our output to be displayed at.

For example, what if we wanted to display the username aligned to the left, then display a (:) and display the user's ID (which is the 3rd field).

We can do that using the following command.

Notice how we replaced the print function with a printf function

```
  GNU nano 2.8.7                                          File: script.awk
awk -F ":" ' NR==2,NR==10 { printf "%-8s : %3d\n", $1,$3 } ' /etc/passwd
```

The printf function takes the format which we want to display our output in.

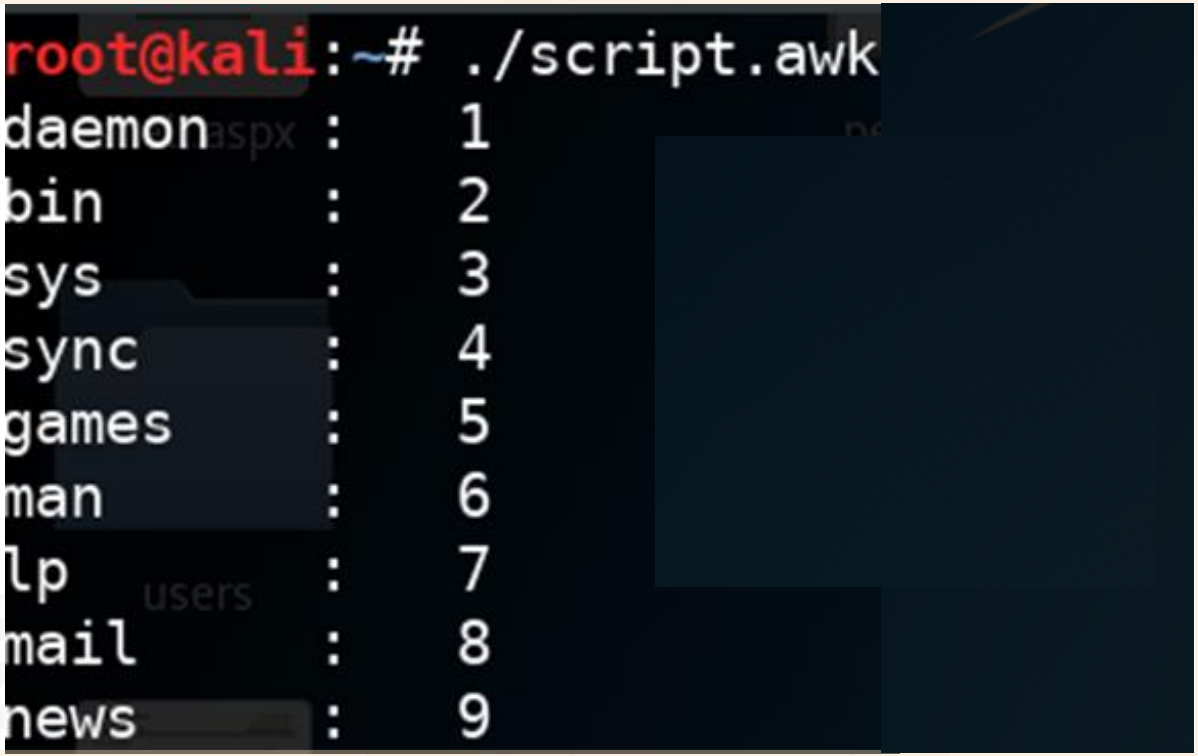
In the previous command, we're telling awk that we want to print two fields (each denoted by an % sign) and we want print a : between them.

The first field is justified to the left (hence the – sign on the left) and it's 8 digits long.

The second field is three digits long and it's justified to the right and followed by a new line (the \n symbol).

8.3 Using Linux Tools for Log Analysis

Notice that awk reserved space for eight characters regardless of the size of the field.



Finally, we can use the **begin** keyword to specify the command that would only get executed once. For example, if you want to display a header before displaying the content of our file we can do so in the beginning section.

```
  GNU nano 2.8.7                File: script.awk
awk -F ":" '
BEGIN { printf "%-8s : %-6s\n","Uname","UserID"  }
NR==1,NR==10 { printf " %-8s : %3d\n", $1,$3 } ' /etc/passwd
```

Web Logs

8.4 Web Logs

This section covers the analysis and examination of logs produced by the web servers.

We will examine two famous web servers and their logs.

Apache web server, which is usually used with Linux distributions, and the **IIS** which is Microsoft operating systems default web server.

When analyzing web servers logs, there is good news and bad news.

The good news is that web servers implements text logs and document the events in a relatively simple format.

The bad news is that due to their nature and the huge amount of requests per second, it wouldn't be feasible to just open a web server log/event file in a text editor and search for something.

Different web servers have different formats to log events.

Understanding that format is important for parsing the log file.

Since we probably can't read the log line by line, we're going to need to have a tool that does that for us.

Apache HTTP Server, is a free open-source cross-platform web server software.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

Although Apache HTTP Server is cross-platform, as of 1 June 2017, 92% of all Apache HTTPS Server copies run on Linux distributions.

Version 2.0 improved support for non-Unix operating systems, such as Windows and OS/2.

On a Linux machine, you can usually find apache logs in the **/var/www/apache2/logs** directory.

The one we're interested at is the **access logs**. Which contains the requests made by the clients to the server and their IP address.

Note: Apache logs can also be found on Windows in the apache directory.

This is an example of an apache log entry.

```
10.100.5.170 - - [16/Oct/2017:13:31:03 +0000] "GET /application/example.php?name=Parameter HTTP/1.1" 200 898
"http://10.100.0.28/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36"
```

8.4 Web Logs

The log starts with the client's IP address, followed by the date and the request the it made.

Next, we can see the status code which the server generated for that request (200 means that the requested resource was found).

```
10.100.5.170 - - [16/Oct/2017:13:31:03 +0000] "GET /application/example.php?name=Parameter HTTP/1.1" 200 898
"http://10.100.0.28/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36"
```

The size of the request (898 in this example) is stated after the response code.

At the end we have the webserver's IP address (the requested host) and the user-agent or the client's browser.

```
10.100.5.170 - - [16/Oct/2017:13:31:03 +0000] "GET /application/example.php?name=Parameter HTTP/1.1" 200 898
"http://10.100.0.28/" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100
Safari/537.36"
```

As mentioned earlier, we can view this file (with the right privileges) using a text editor.

However, since it would be really painful to sort the results or look for a specific entry based on the text editor alone, we're going to need the help of tools we explained earlier to make the process easier.

If we want to manually inspect the logs for attacks, we should have a fair understanding for the web attacks and their payloads.

For example, Cross Site Scripting payloads are usually associated with the use of **<script>** tags.

8.4 Web Logs

MAP REF

```
root@debian:/var/log/apache2# cat access.log | grep script
192.168.1.2 - - [22/Oct/2017:00:56:49 +0000] "GET /comment.php?name=%3Cscript%3E
alert(%27xss%20attack%27)%3C/script%3E HTTP/1.1" 200 840 "-" "Mozilla/5.0 (Windo
ws NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.1
00 Safari/537.36"
root@debian:/var/log/apache2# _
```

SQL injection attacks usually involves sending SQL queries or uses conjunction keywords such as AND/OR within the HTTP request.

We can search for those attacks within the log file by grepping for SQL keywords or for single quotes which are also commonly used in SQLi attacks.

Note: quotes are usually encoded as **%27** using URL encoding.

8.4 Web Logs

MAP REF

```
root@debian:/var/log/apache2# cat access.log | grep or
192.168.1.2 - - [22/Oct/2017:00:57:03 +0000] "GET /store/viewproducts.php?name=r
oot%27%20or%20%271%27=%271 HTTP/1.1" 200 984 "-" "Mozilla/5.0 (Windows NT 6.1; k
in64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/53
7.36"
root@debian:/var/log/apache2# _
```

Command injection attacks usually involve executing netcat with something like “**nc –lvp 5555 –e /bin/bash**”.

Directory traversal attacks involve sending requests that contains many “../” and some OS default file like “**/etc/passwd**”.

Apache allows the administrator to get a more detailed logging when normal logging isn't enough.

This is can be done by adding the **mod_log_forensic** module to the server.

This part of the module provides forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request.

The first time is **before** it's processed further, that is, after receiving the headers. The second log entry is written **after** the request processing at the same time where normal logging occurs[5].

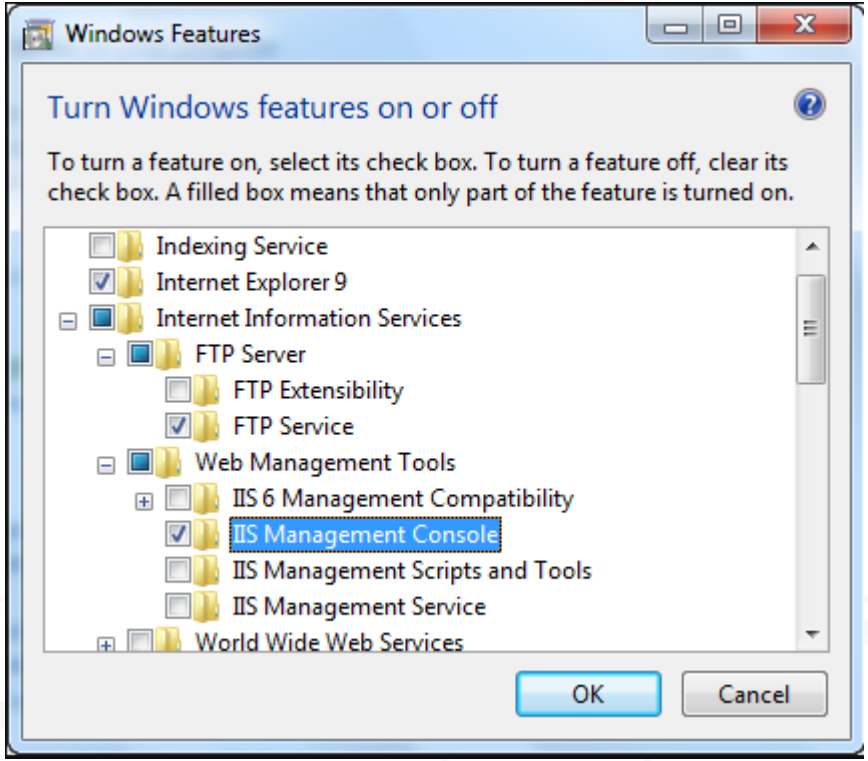
Internet Information Services (**IIS**) is an extensible web server created by Microsoft for use with the Windows NT family.

IIS supports HTTP, HTTPS, FTP, FTPS, SMTP and NNTP.

IIS has been an integral part of the Windows NT family since Windows NT 4.0, though it may be absent from some editions (e.g. Windows XP Home edition), and is not active by default.

Unlike many other products, IIS logging is usually enabled by default. However, it is not uncommon to find logging disabled on some Windows Installations.

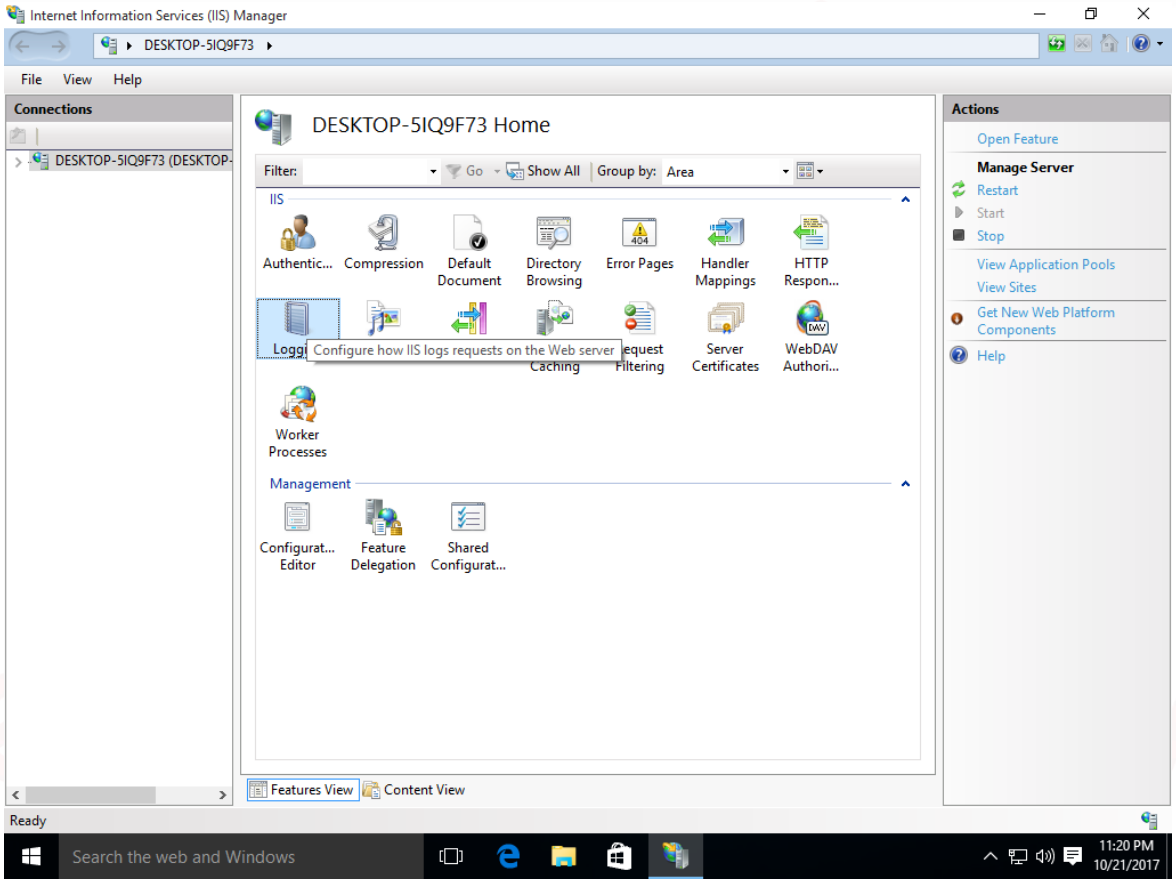
IIS logs can be viewed from the built-in management console in Windows. Note that sometimes, you'll need to install the management console after installing IIS.



The management console can be launched by typing **Internet Information Services** in the start menu.

Once the console is running, the logging configuration can be viewed an edited from the **logging** icon.

8.4 Web Logs



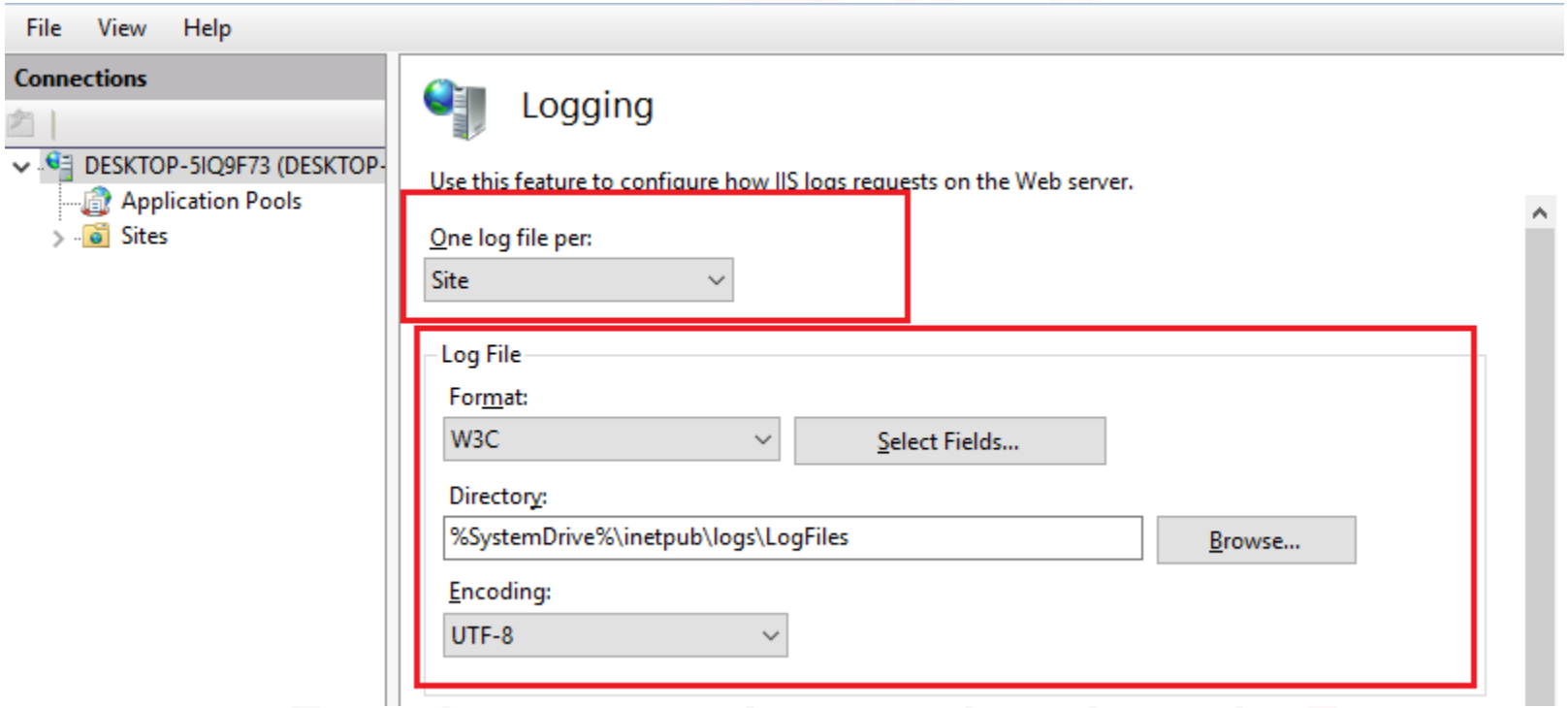
IIS uses the **W3C** format to store the records.

The W3C, also known as the extended log file format, has many fields which the administrator can choose to include or exclude from the logs.

By clicking on the **logging** icon, we can view the edit the logging configuration.

8.4 Web Logs

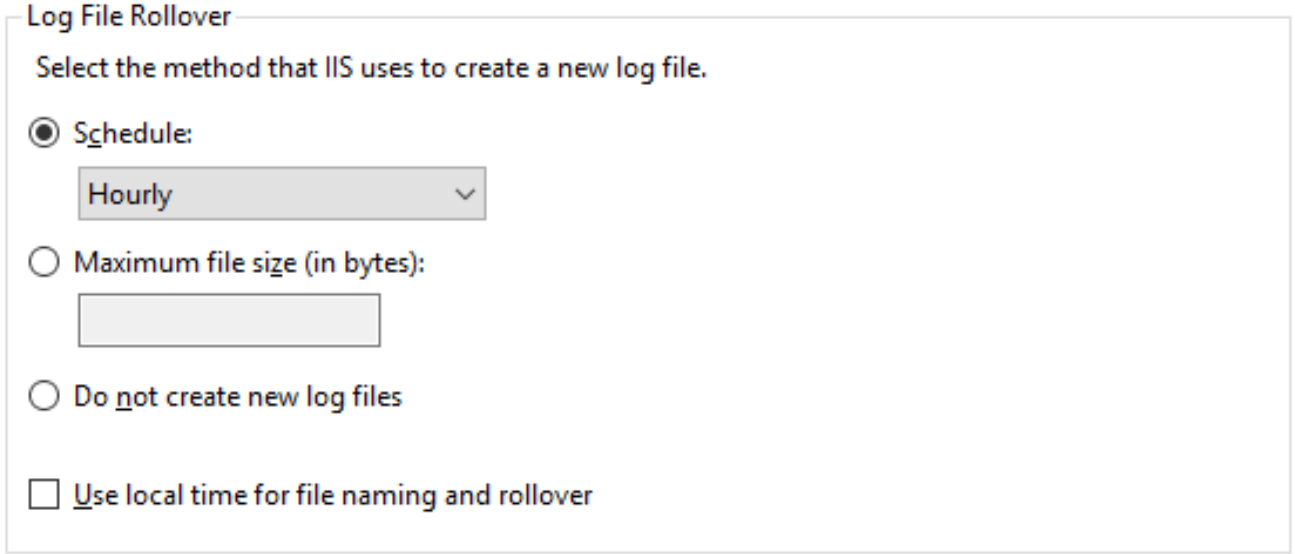
MAP REF



The logging configuration specifies the number of log files used (per server or per site) and the path where the logs are stored.

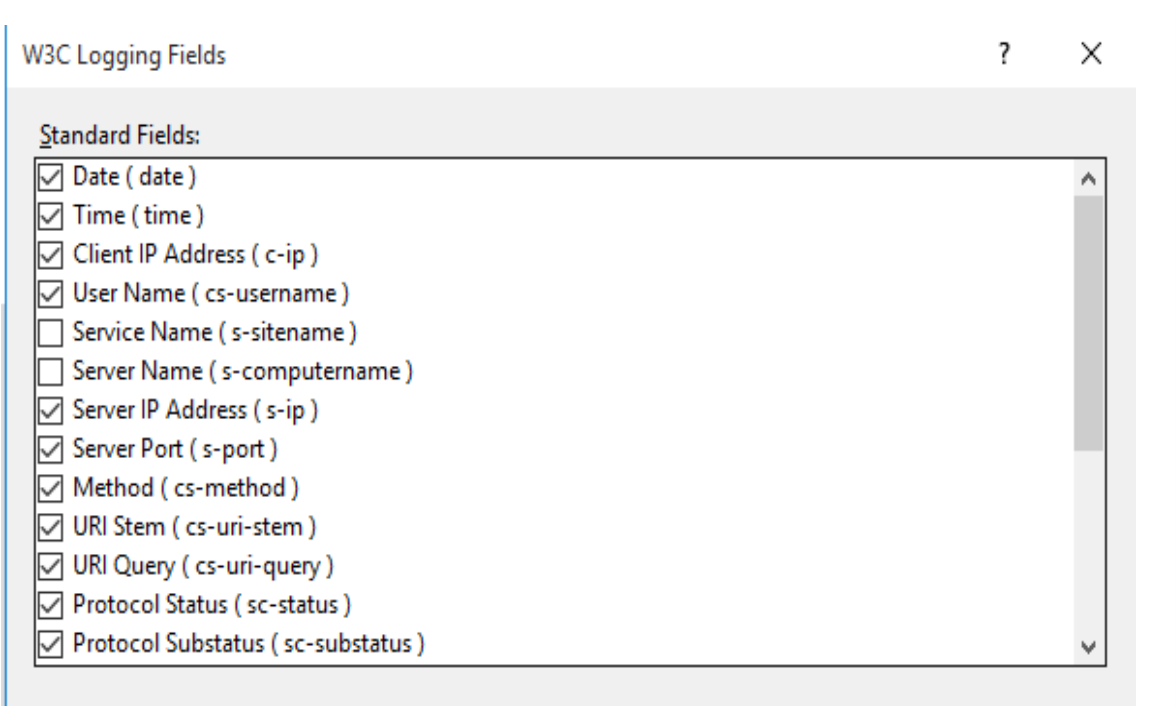
The console allows the user to specify how and when new log files are to be created (every period of time or after the old log reaches a certain size).

Note: the console uses the GMT time zone unless it was told to use the local time zone.

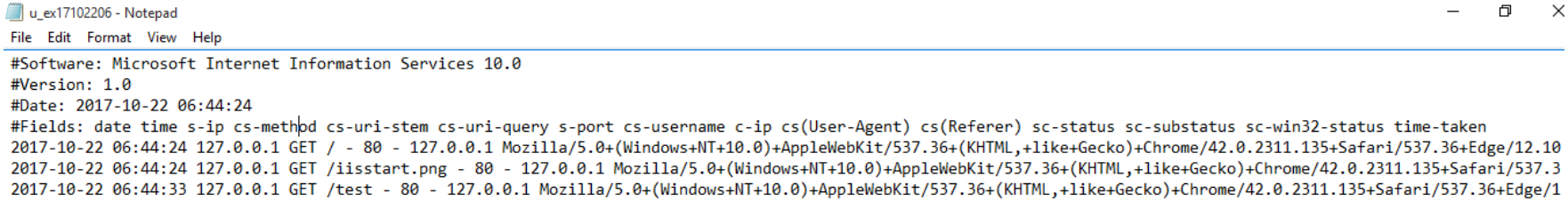


The Window also specifies the encoding used to store the log records.

Additionally, it allows the administrator to select the fields which they want to be included in the log.



The log file of IIS is very neat. It starts with the server's header which includes the server's signature and date followed by a description and order of the log's records, which makes it easier to read, understand and parse.



Windows Events

8.5 Windows Events

Windows is obsessed about logging the details; it has different categories for different records and refers to its log files as **event logs**.

Event logs are used to store useful information regarding events that occurred on the system and their users.

The events logged by the operating system gets written into one of four separate files: **Applications file , Security file, Hardware and System file.**

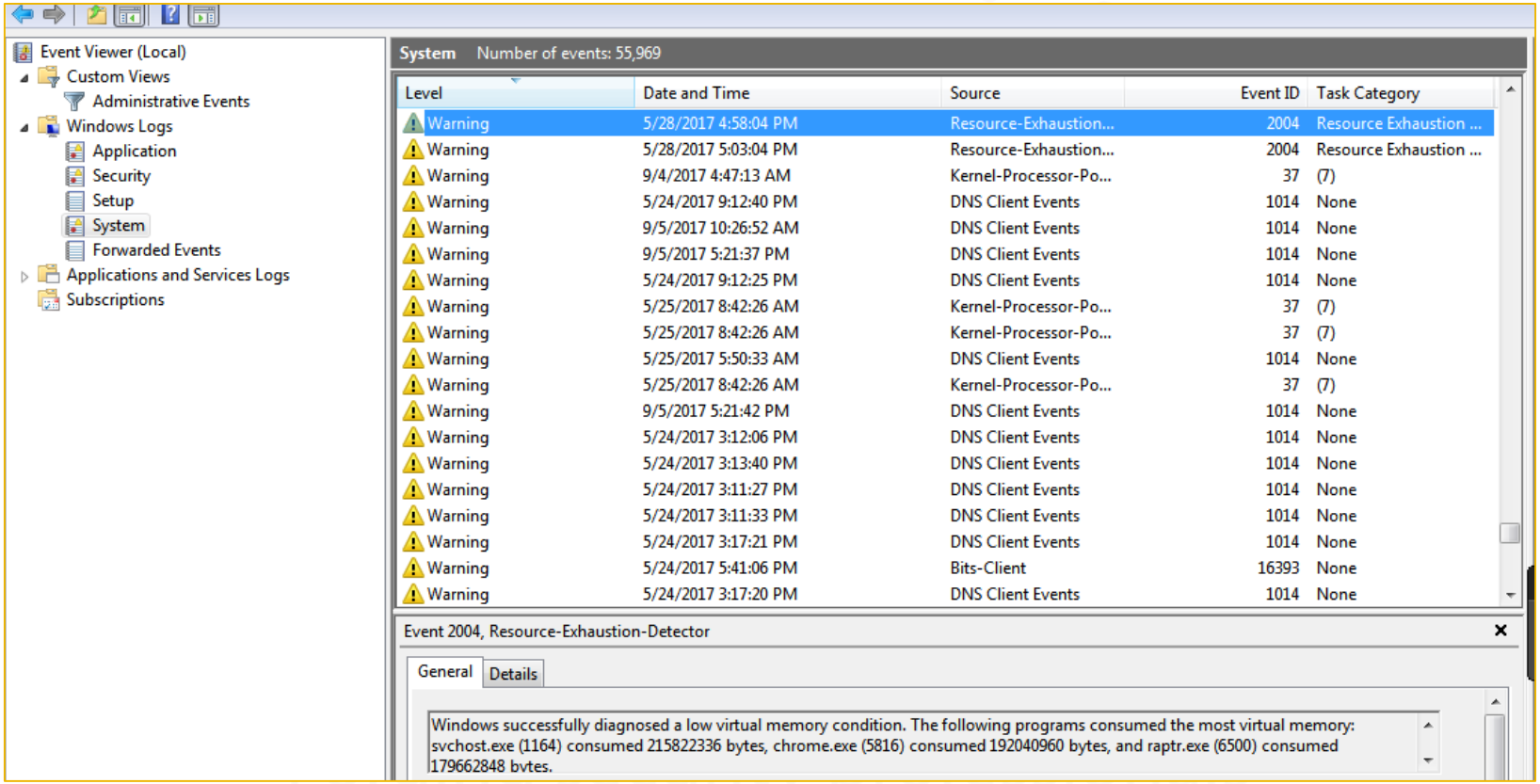
Each event has an ID number which can be queried using it.

System file stores general events regarding the operating system itself.

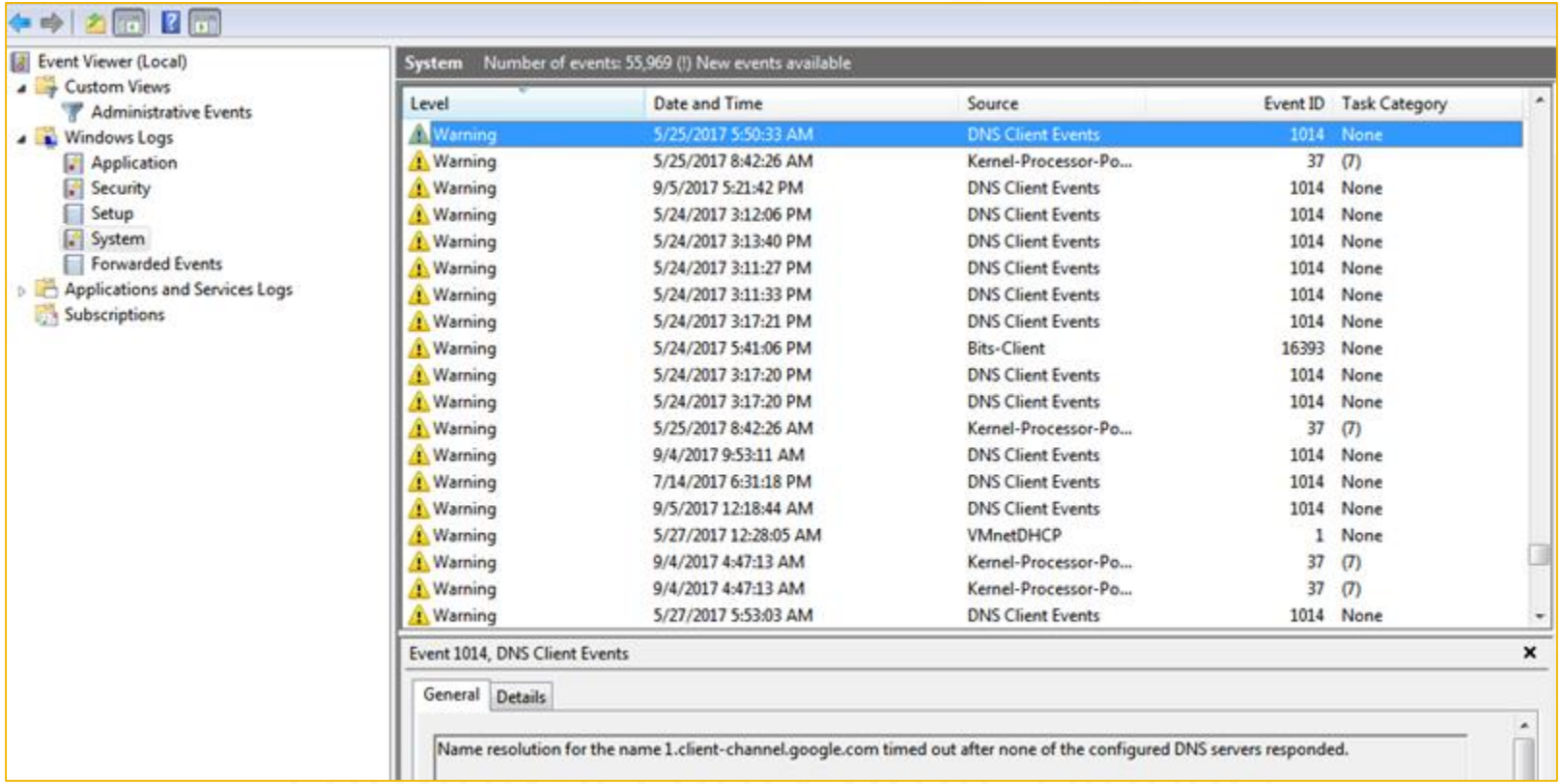
For example, in the upcoming screenshots is logging a warning regarding memory shortage and failed DNS request respectively.

8.5 Windows Events

MAP REF

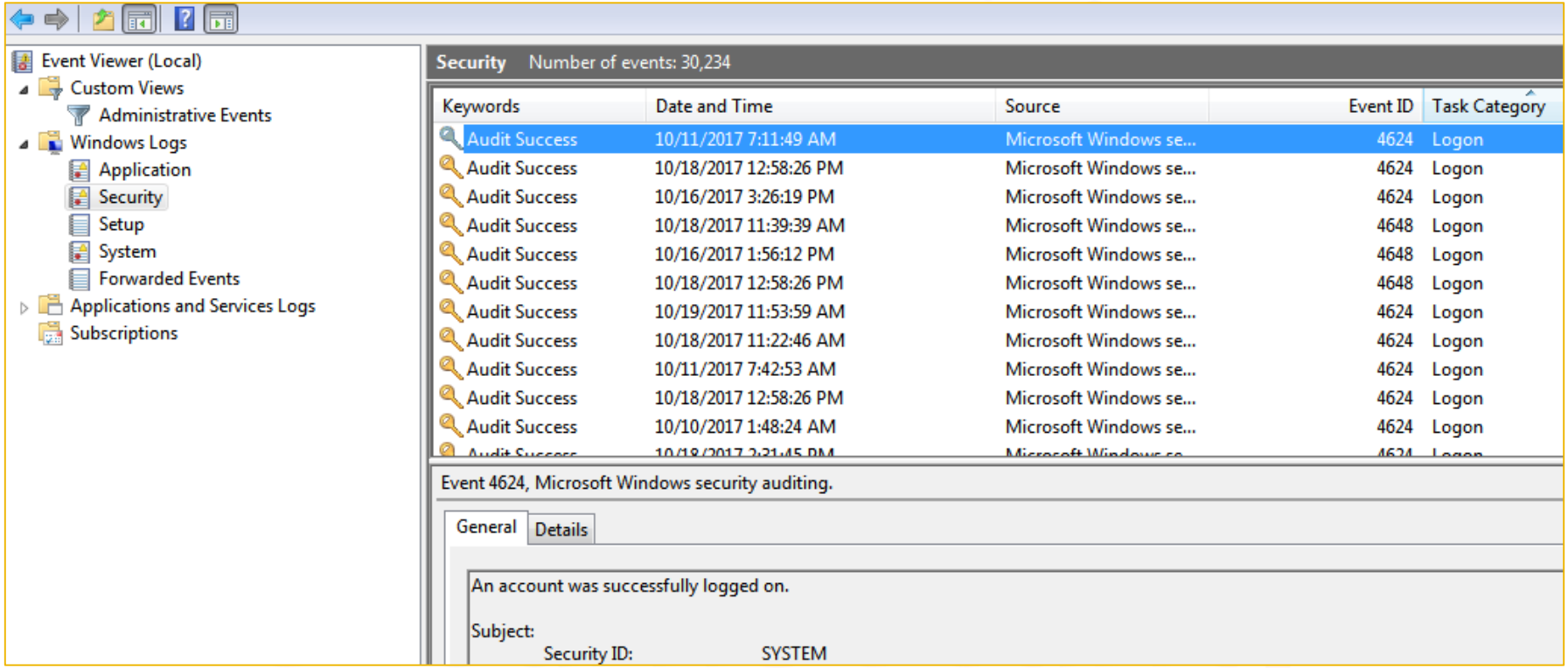


8.5 Windows Events



The security files logs events related to security processes such as login attempts.

8.5 Windows Events



There are many differences between Windows logging mechanism used in the systems before Windows Vista and after.

One of the main differences is in the storage location, as XP and server 2003 stores their event in a ***.evt** files at **system32/config** directory.

While systems that came after XP (Vista and later versions) store their events on a ***.evtx** files at the **system32\winevt\Logs** directory.

Note that the files are NOT text files. They are stored in a preparatory binary format from Microsoft.

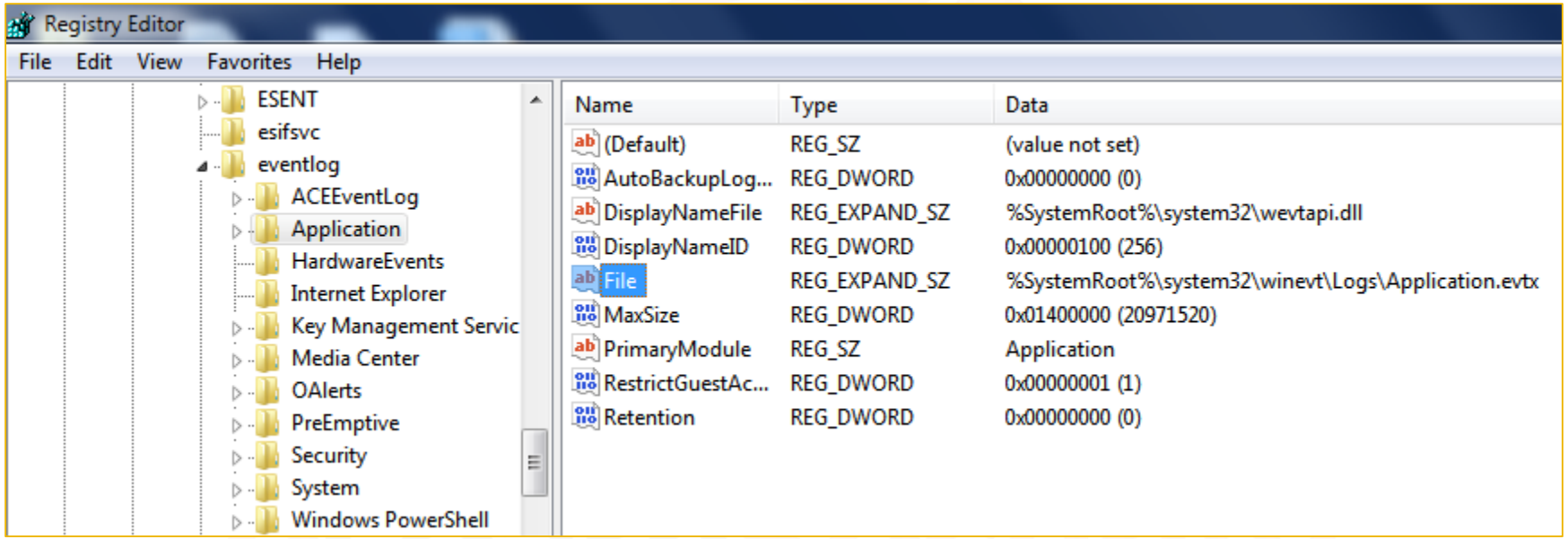
The default path for storing the log files can be changed from the registry.

Each file has its own entry which it can be modified from **HKLM\SYSTEM\CurrentControlSet\services\eventlog** Path.

For example, we can change the default path for storing **application** related logs at :

- **HKLM\SYSTEM\CurrentControlSet\services\eventlog\Application**

8.5 Windows Events



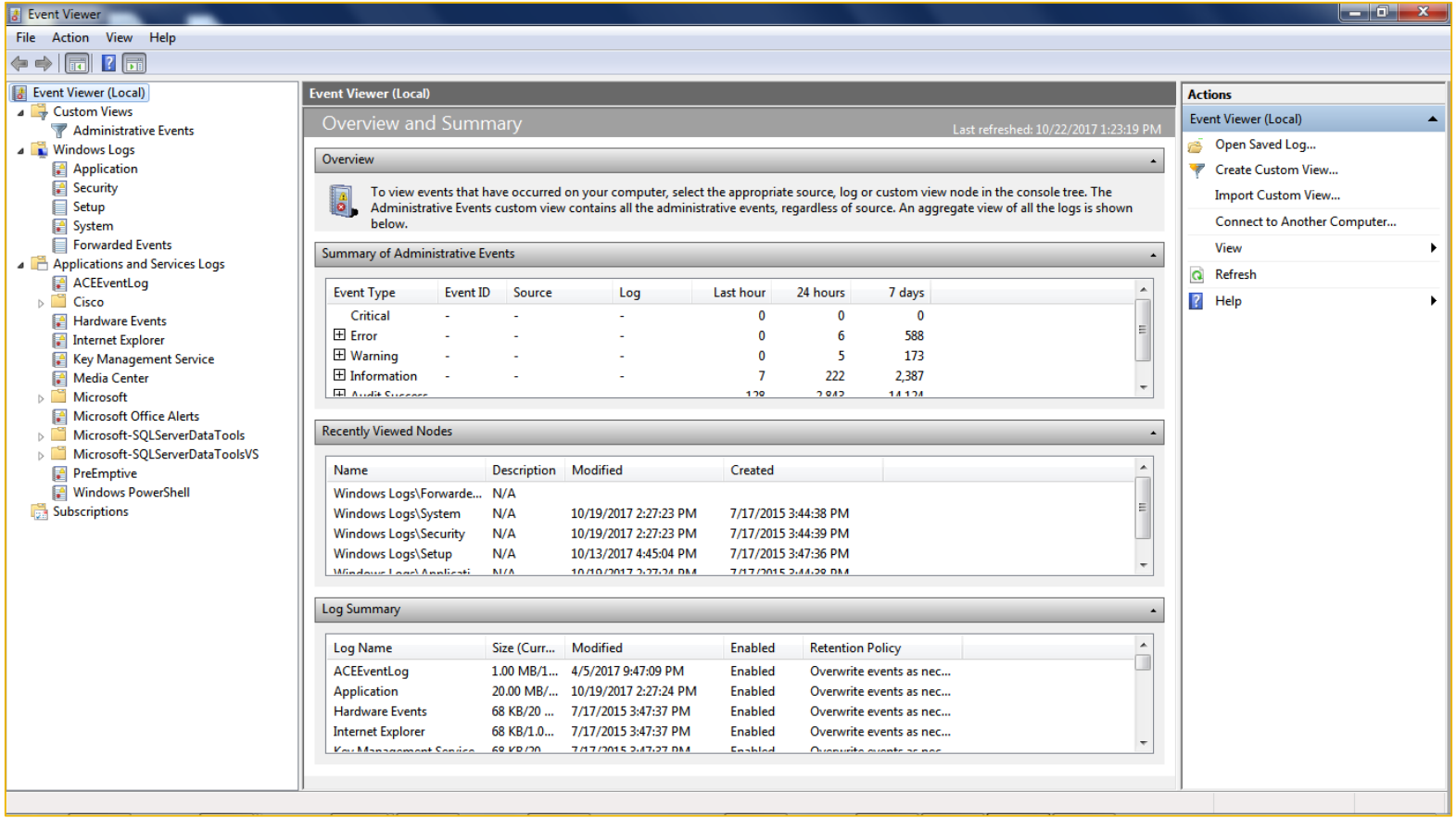
Since the logs aren't stored in text format, we're going to need a special tool to view them.

We can use the **Microsoft Event Viewer** to examine those logs.

Just type event in the start menu search bar and choose the event viewer icon.

8.5 Windows Events

MAP REF



The other difference is the numbering system used by the logging system in Windows XP and Windows Vista & later versions.

For example, the logoff event's ID in windows XP is 551, while on Vista and later versions it's 4647.

Even though some of Windows XP events can be converted to Windows 7/8 events by adding 4096 to the event's ID, the two systems aren't fully compatible as Windows 7/8 introduced new events that didn't exist in windows XP.

Also, it is worth mentioning that Servers editions of Windows logs more events than normal desktop installations.

Windows tends to log fewer events if not told by the administrator.

One very important decisions an administrator (or a threat hunter) must make, is to decide which events should be logged.

Although it may be tempting to log everything, that approach may cause more damage than good, as it might make the analysis harder and slower.

8.5 Windows Events

MAP REF

The are few categories of events that are essential for forensics investigations. Those types are:

1. Application Whitelisting
2. Application Crashes
3. Clearing Event Logs
4. Software and Service Installation
5. Account Usage
6. External Media Detection
7. Lateral Movement Detection

8.5 Windows Events

MAP REF

Windows XP Old Event ID	Windows 7/8 Event ID	Description
528	4624	Successful Login
529	4625	Failed Login attempt
680	4776	Successful Account Authentication
624	4720	Creating of a new user
636	4732	A member has been added to a local group
632	4728	Member has been added to a global group.
2949	7045	Service Creation

A more detailed list about the Windows Security Related events can be found on the NSA's website.

https://www.iad.gov/iad/library/reports/spotting-the-adversary-with-windows-event-log-monitoring.cfm

Windows Vista and above Events

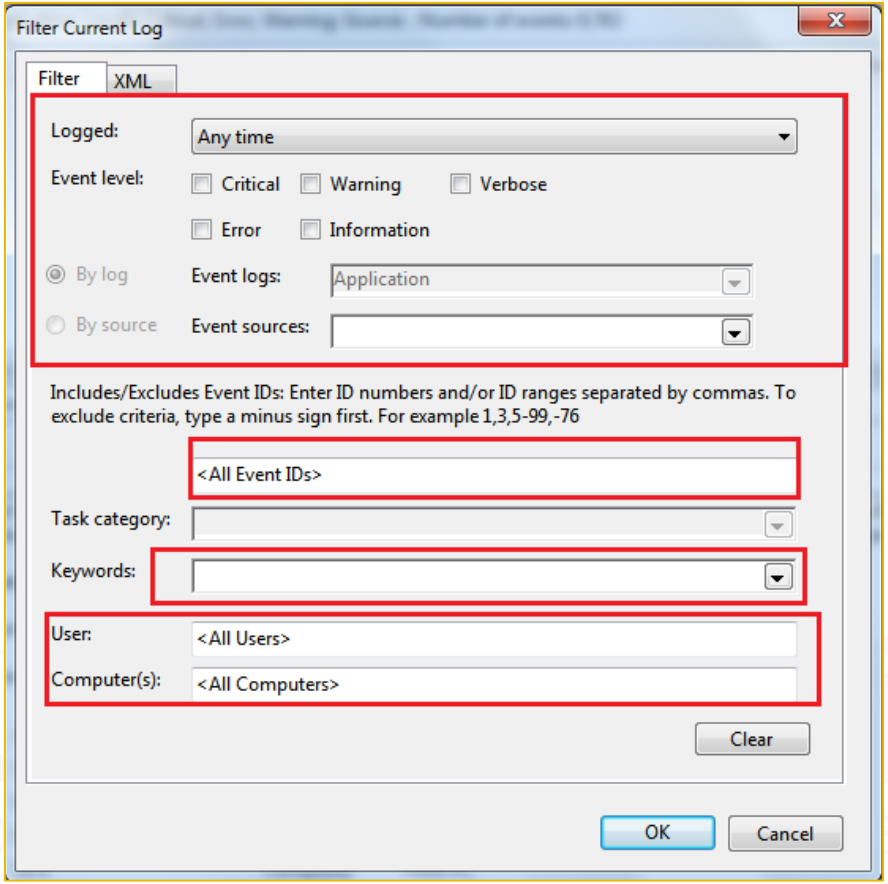
General Event Descriptions	General Event IDs
Account and Group Activities	4624, 4625, 4648, 4728, 4732, 4634, 4735,4740, 4756
Application Crashes and Hangs	1000 and 1002
Windows Error Reporting	1001
Blue Screen of Death (BSOD)	1001
Windows Defender Errors	1005, 1006, 1008, 1010, 2001, 2003, 2004, 3002, 5008
Windows Integrity Errors	3001, 3002, 3003, 3004, 3010 and 3023
EMET Crash Logs	1 and 2
Windows Firewall Logs	2004, 2005, 2006, 2009, 2033
MSI Packages Installed	1022 and 1033
Windows Update Installed	2 and 19
Windows Service Manager Errors	7022, 7023, 7024, 7026, 7031, 7032, 7034
Group Policy Errors	1125, 1127, 1129
AppLocker and SRP Logs	865, 866, 867, 868, 882, 8003, 8004, 8006, 8007
Windows Update Errors	20, 24, 25, 31, 34, 35
Hotpatching Error	1009
Kernel Driver and Kernel Driver Signing Errors	5038, 6281, 219
Log Clearing	104 and 1102
Kernel Filter Driver	6
Windows Service Installed	7045
Program Inventory	800, 903, 904, 905, 906, 907, 908
Wireless Activities	8000, 8001, 8002, 8003, 8011, 10000, 10001, 11000, 11001, 11002, 11004, 11005, 11006, 11010, 12011, 12012, 12013
USB Activities	43, 400, 410
Printing Activities	307

Table 1: Vista and above Events

Event Viewer allows us to filter out the log files and only display the records that match our search criteria.

This can be done by pressing on the filter current log button and choosing the filter criteria.

Using the filtering option we are allowed to filter out the results based on the event severity level, time, event ID, keyword or user account.



Syslog

If you're familiar with router or switch configuration, then you've probably seen a message like this before.

```
Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0,
changed state to up
```

These messages are usually displayed by the networking device (router or switch) on the configuration terminal to give the administrator a message.

Depending on the type of the message, it can be tell us anything from interface status change all the way to fatal errors.

What will happen if, for example, you were an administrator in a network that is made up of 5 switches and 2 routers?

Following all those messages on 7 different networking devices would not be feasible at all.

Not to mention that those messages may appear sometime after the working hours and disappear (because of the arrival of other messages for example).

This is where **syslog** comes to our aid.

Syslog is a standard for message logging. It allows separation of the software that generates messages, the system that stores them, and the software that reports and analyzes them.

Each message is labeled with a facility code, indicating the software type generating the message and assigns a severity label [6].

Syslog can be enabled on networking devices and clients. It allows the administrator to group networking messages on one server for later analysis and examination.

Depending on the product, some systems can only store the logs for manual analysis by the Administrator, while others come with a built in analyzers that analyzes the messages on the fly.

The **facility** is used to describe the origin of that message.

In other word, what type of software/hardware issued that message.

The following tables provides a brief description for some of the facility's codes.

Facility Code	Type
0	Kernel Messages.
1	User Messages.
2	Mail system
3	System Services (daemons)
4	Authentication messages.

As the name suggests, the **severity** indicates the level of the message's sensitivity.

Message Code	Message Severity
0	Emergency
1	Alert
2	Critical
3	Error
4	Warning
5	Notice
6	Informational
7	Debug

It is important to remember that when specifying a severity level, the syslog client will send messages from the specified level and above.

For example, if the administrator set the severity level to 4, the client will send messages from level 0 to 4.

On Cisco routers, Syslog can be enabled by specifying the server's IP address and severity level.

```
Router(config)#logging 192.168.1.1
Router(config)#logging trap debugging
Router(config)#logging on
Router(config)#
```

Log Analysis using Linux

How to use Linux commands to analyze different logs.

References

[1] en.wikipedia.org/wiki/Log_analysis
[2] https://en.wikipedia.org/wiki/AWK
[3] https://en.wikipedia.org/wiki/Apache_HTTP_Server
[4] https://en.wikipedia.org/wiki/Internet_Information_Services
[5] https://httpd.apache.org/docs/2.4/mod/mod_log_forensic.html
[6] https://en.wikipedia.org/wiki/Syslog

References

Secunia

Vulnerability review

Immunity Debugger

WinDBG