Windows Event Log Format

2005-07-10 - I've recieved a lot of positive feedback over the last year and a half on the work presented below. One recent email I received was from a fellow named Tim who has written an opensource parser for Windows Event Logs. I have not used them yet but those looking for a solution on xNIX platforms might want to give these (Python based) scripts a try. Cheers and and thanks for working on this and contacting me Tim.

http://www.sentinelchicken.org/projects/grokevt/

##################################

Prologue: Microsoft Windows(c) logs can provide a wealth of information when troubleshooting or conducting analysis of an event of interest. Unfortunately its pretty hard to find documentation on the native format of these logs which serves more to hinder security interests rather than foster better ones and better auditing.

Many readers may be wondering why is it necessary to know the format of these logs when "I have my handy 'Event Viewer' that will display the information to me when I need it". Others still might say "Yeah but Microsoft has released tools like 'Log Parser'* which will rumble through the logs and output the information I want in a number of different formats". Still others yet may say "I see a CPAN PERL Module for working with Windows Event Logs (Win32::Event) - why not use it?". While still others might say "You can go and purchase wonderful commercial offerings like GFI S.E.L.M. that can do the job for me". The answer is all of these tools require a Windows Operating System to run.

This is all fine until you decide to do some forensics on a compromised system and cannot boot up windows to launch your favourite Win32 application. Sure, you can still copy the evidence AppEvent.Evt and relevant event files to another post analysis Windows system for processing but for a large percentage of the forensics field this introduces more complications and requires more resources be at the disposal of the forensic investigator on top of their regular toolkit - usually running on a UNIX based machine.

Now that we understand why someone would want to know how to read these log files in their native format on an operating system other than Windows we'll take a look at the binary format of these logs and attempt to break out the fields. I have not managed to break out every field yet but in all honesty I have not devoted more than a few evenings to the task and I have not verified this information with external sources (its hard to find any...).

I've found a few docs on Event Log format

http://www.codeproject.com/system/sysevent.asp
http://leb.net/wine/WinDoc/msdn/sdk/platforms/doc/sdk/win32/struc/src/str07_8.htm

Here is a copy from the codeproject of the format. I originally found it in a PDF file but Google isn't returning it in my search strings any more and I don't have the URL available now.

```
typedef struct _EVENTLOGRECORD {
DWORD Length;
DWORD Reserved;
DWORD RecordNumber;
```

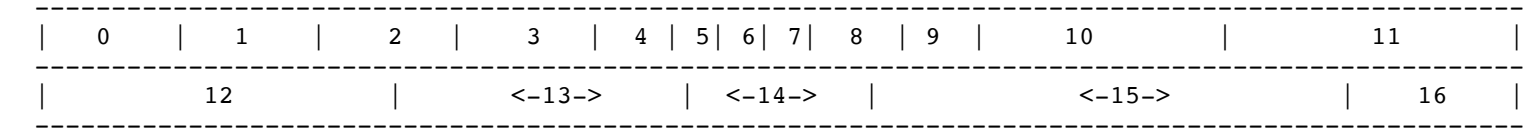
```
DWORD TimeGenerated;
DWORD TimeWritten;
DWORD EventID;
WORD EventType;
WORD NumStrings;
WORD EventCategory;
WORD ReservedFlags;
DWORD ClosingRecordNumber;
DWORD StringOffset;
DWORD UserSidLength;
DWORD UserSidOffset;
DWORD DataLength;
DWORD DataOffset;
//
// Then follow:
//
// TCHAR SourceName[]
// TCHAR Computername[]
// SID UserSid
// TCHAR Strings[]
// BYTE Data[]
// CHAR Pad[]
// DWORD Length;
//

}
```

Below is what I imagined comprised an eventlog entry. I'll update this and the script I wrote when I have more time.

See http://www.whitehats.ca/downloads/malik/evt_log_parse.txt for a PHP script to parse through event logs

######

So... here is what I believe comprises a Windows Event Log in binary format:



0: Message Separator (4 bytes - Separator is (10011000110011001001100011001101 - bin) or (4c664c65 - hex) or (LfLe - ascii))
1: Message Number (4 bytes - Padded with null characters from left to right to fill in the full 4 bytes. It is in little endian byte order)
2: Date Created (4 bytes, little endian, decimal value in epoch)
3: Date Written (4 bytes, little endian, decimal value in epoch)
4: Event ID (2 bytes, little endian, decimal value)
5: Unknown? (1 byte)
6: Unknown? (1 byte)
7: Event Type (1 byte - number - used as index to retrieve 'Event Name')
8: String Count (2 bytes - The number of strings in the event in decimal)
9: Category (2 bytes - decimal value)

10: SID? (8 bytes - possibly the decimal value of the SID)
11: Unknown? (8 bytes - possibly related to the SID)
12: Unknown? (11 bytes)
13: Source Name (Variable Length in words (4 bytes) with at least the last two bytes being null and 0 or more bytes of null padding up to the length of a full word)
14: Computer Name (Variable Length in words (4 bytes) with at least the last two bytes being null and 0 or more bytes of null padding up to the length of a full word)
15: String1 (Variable Length - Also known as the 'Message' - Variable Length in words (4 bytes) with at least the last two bytes being null and 0 or more bytes of null padding up to the length of a full word.)
16: String'*n*' (Depending on the number of strings specified by
17: Unknown? (8 bytes)

End of event log: hex string noted: '11111111222222223333333344444444' - could be an end marker

When I have more info or a working script to parse through these logs I'll update this page and include the new information. If you can verify any of this information, correct, or fill in the blanks please drop me an email.

Malik

* - Log Parser 2.1 with the IIS6 Resource Kit requires you have WindowsXP or Windows2003 before the program will allow you to extract the utilities within (as of 2004-02-06).

Other interesting EventLog links:

http://www.eventid.net/
http://66.200.20.8/index.asp

A few Forensic links:

http://www.opensourceforensics.org/tools/unix.html
http://www.knoppix.org/

Non-Active Sitemap