Collecting Windows Security Audit Log data with NXLog and Sysmon

Contents

The Windows Security log already contains a lot of audit data such as login/logon activity, account changes, audit policy changes and other events. Recording of additional security related events such as file access and modification can be turned on by altering the security audit policy.

The list of security audit events are published by Microsoft and can be downloaded as a spreadsheet:

- Security Audit Events for Windows 7 and Windows Server 2008 R2
- Windows 8 and Windows Server 2012 Security Events

Sysmon is a tool available from Sysinternals/Microsoft free of charge and is part of the Windows SysInternals tools. It was first released in August 2014 and version v3.0 came out recently on April 20, 2015. Sysmon is and advanced background monitor that can record additional security-related events for use in intrusion detection and forensics such as:

- Process creation with full command line,
- Loading of drivers,
- Network connections,
- Changes to file creation time.

Sysmon is a simple command line tool, the service can be installed running Sysmon from the command prompt. Below is the Sysmon help output:

```
---------------------------------------------------------------------------
Sysinternals Sysmon v3.00 - System activity monitor
Copyright (C) 2014-2015 Mark Russinovich and Thomas Garnier
Sysinternals - www.sysinternals.com

Usage:
Install:    Sysmon.exe -i <configfile>
              [-h <[sha1|md5|sha256|imphash|*],...>] [-n (<process,...>)]
              [-l (<process,...>)]
Configure:  Sysmon.exe -c <configfile>
              [--|[-h <[sha1|md5|sha256|imphash|*],...>] [-n (<process,...>)]
              [-l (<process,...>)]]
Uninstall:  Sysmon.exe -u
  -c   Update configuration of an installed Sysmon driver or dump the
       current configuration if no other argument is provided. Optionally
       take a configuration file.
  -h   Specify the hash algorithms used for image identification (default
       is SHA1). It supports multiple algorithms at the same time.
       Configuration entry: HashAlgorithms.
  -i   Install service and driver. Optionally take a configuration file.
  -l   Log loading of modules. Optionally take a list of processes to track.
  -m   Install the event manifest (done on service install as well).
  -n   Log network connections. Optionally take a list of processes to track.
  -u   Uninstall service and driver.

The service logs events immediately and the driver installs as a boot-start
driver to capture activity from early in the boot that the service will
write to the event log when it starts.

On Vista and higher, events are stored in
"Applications and Services Logs/Microsoft/Windows/Sysmon/Operational"
On older systems, events are written to the System event log.

If you need more information on configuration files, use the '-? config'
command. More examples are available on the Sysinternals website.

Specify -accepteula to automatically accept the EULA on installation, otherwise
```

```
you will be interactively prompted to accept it.

Neither install nor uninstall requires a reboot.
--------------------------------------------------------------------------------
```

Once configured and installed on a system, Sysmon will remain resident as a windows service to log system activity to the Windows event log. This audit log data can be collected and forwarded by NXLog to SIEM and log analytics systems to allow us to identify malicious or anomalous activity and understand how intruders and malware operate on our network.

On Vista and higher, events are stored in "Applications and Services Logs/Microsoft/Windows/Sysmon/Operational". On older systems, events are written to the System event log.

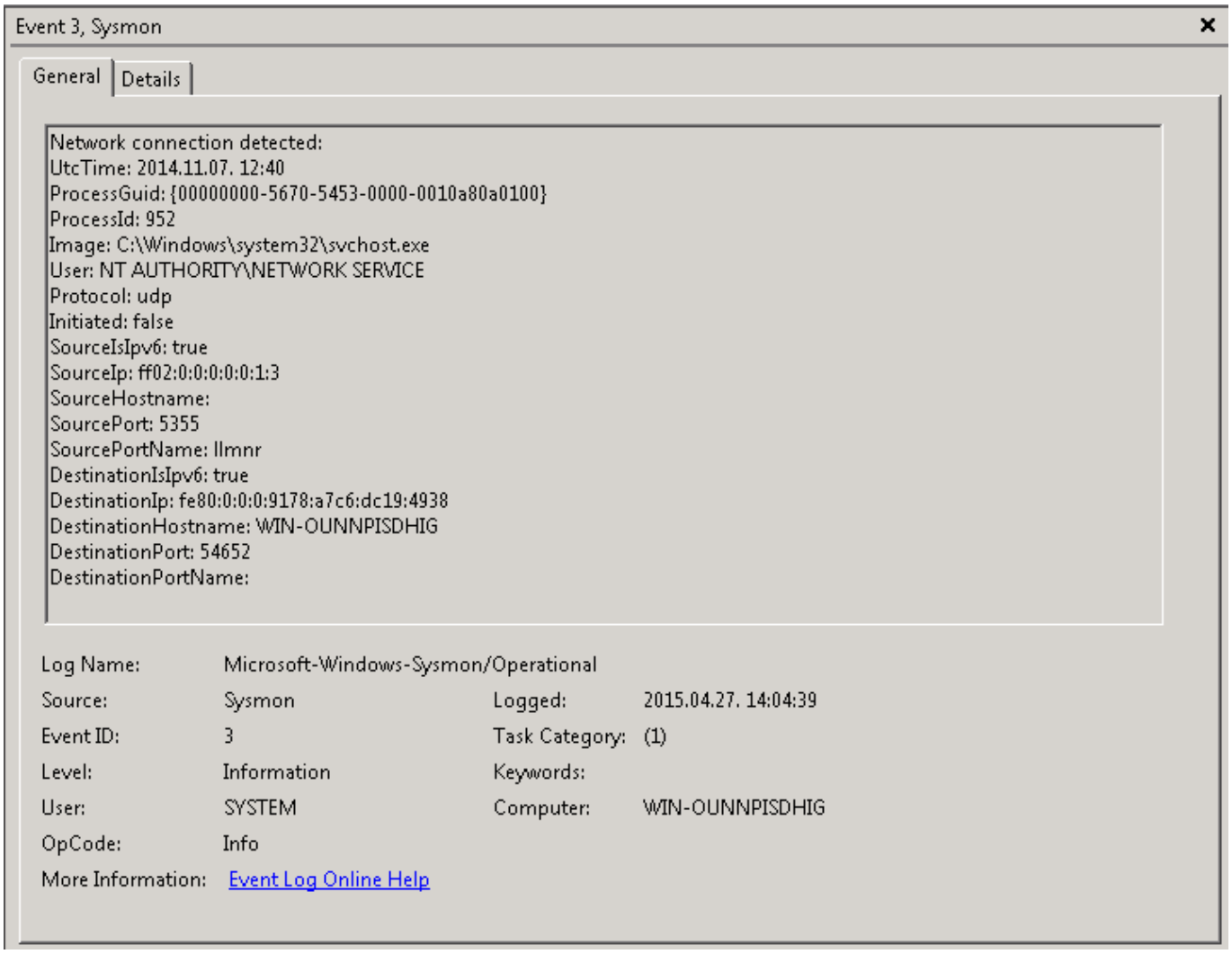

Figure 1: An Eventlog record generated by Sysmon

Sysmon can be quite noisy and we may not want to record all the events it logs. An XML configuration file can be used with the `-c` or `-i` command line switch to fine-tune the behavior of Sysmon. Below is an XML configuration example:

```
<Sysmon schemaversion="1.0">
<Configuration>
  <!-- Capture MD5 Hashes -->
  <Hashing>MD5</Hashing>
  <!-- Enable network logging -->
  <Network />
```

```
</Configuration>
<Rules>
    <!-- Log all drivers except if the signature -->
    <!-- contains Microsoft or Windows -->
    <DriverLoad default="include">
        <Signature condition="contains">microsoft</Signature>
        <Signature condition="contains">windows</Signature>
    </DriverLoad>
    <!-- Do not log process termination -->
    <ProcessTerminate />
    <!-- Exclude certain processes that cause high event volumes -->
    <ProcessCreate default="include">
        <Image condition="contains">noisyprogram.exe</Image>
    </ProcessCreate>
    <!-- Do not log file creation time stamps -->
    <FileCreateTime />
    <!-- Do not log network connections of a certain process or port -->
    <NetworkConnect default="include">
        <Image condition="contains">someapp.exe</Image>
        <DestinationPort>4041</DestinationPort>
    </NetworkConnect>
</Rules>
</Sysmon>
```

The Sysmon manual has more details about the configuration file and the event filtering tags.

Now we will want to collect and ship these logs to our SIEM or log analytics system. This is where NXLog steps in. The following `nxlog.conf` configuration collects Sysmon generated event records from the Windows Eventlog. For testing purposes we use a configuration which only reads Sysmon's events and writes them to a file in JSON format.

```
define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log
LogLevel INFO

<Extension _json>
    Module xm_json
</Extension>

<Input in>
    Module im_msvistalog
    Query <QueryList> <Query Id="0"> <Select Path="Microsoft-Windows-Sysmon/Operational ↩
        ">*</Select> </Query></QueryList>
</Input>

<Output out>
    Module      om_file
    File 'C:\test\sysmon.json'
    Exec to_json();
</Output>

<Route 66>
    Path        in => out
</Route>
```

Sysmon generated events contain event details in a structured format in the *`EventData`* section as shown in the screenshot below.

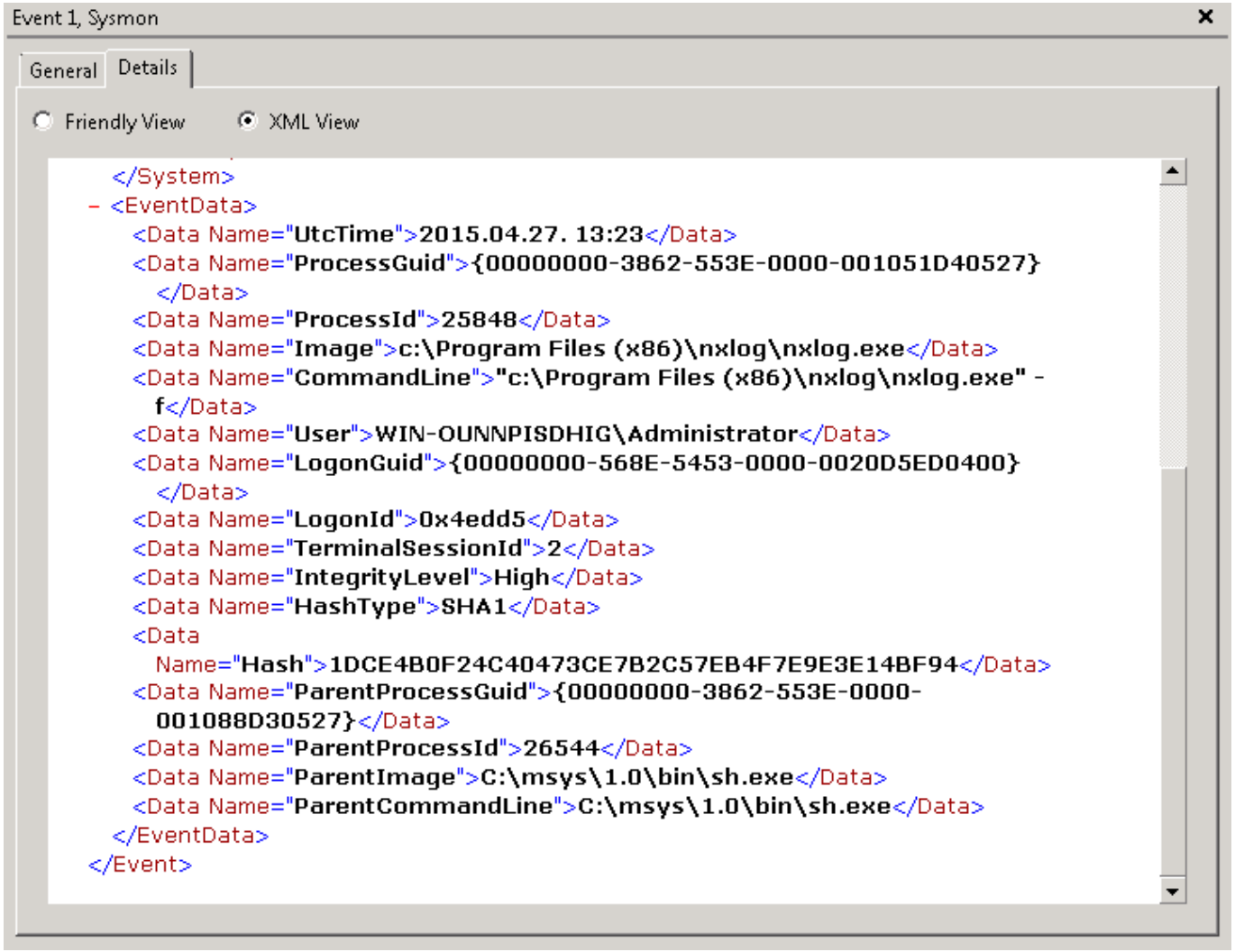

Figure 2: EventData XML fields of a Sysmon generated Eventlog record

This data is recorded in the same structure as the events under the Security log using the `<Data name="key">value</Data>` tags. NXLog will automatically parse this so that these values are accessible as NXLog fields. When converted to JSON with our NXLog configuration, the event record will look as follows. Note that this JSON is pretty-printed for readability, NXLog generates single-line JSON records.


```
{
 "EventTime":"2015-04-27 15:23:46",
 "Hostname":"WIN-OUNNPISDHIG",
 "Keywords":-9223372036854775808,
 "EventType":"INFO",
 "SeverityValue":2,
 "Severity":"INFO",
 "EventID":1,
 "SourceName":"Microsoft-Windows-Sysmon",
 "ProviderGuid":"{5770385F-C22A-43E0-BF4C-06F5698FFBD9}",
 "Version":3,
 "Task":1,
 "OpcodeValue":0,
 "RecordNumber":2335906,
 "ProcessID":1680,
 "ThreadID":1728,
```

```
 "Channel":"Microsoft-Windows-Sysmon/Operational",
 "Domain":"NT AUTHORITY",
 "AccountName":"SYSTEM",
 "UserID":"SYSTEM",
 "AccountType":"Well Known Group",
 "Message":"Process Create:\r\nUtcTime: 2015.04.27. 13:23\r\nProcessGuid:  ↩
    {00000000-3862-553E-0000-001051D40527}\r\nProcessId: 25848\r\nImage: c:\\Program Files  ↩
    (x86)\\nxlog\\nxlog.exe\r\nCommandLine: \"c:\\Program Files (x86)\\nxlog\\nxlog.exe\" - ↩
    f\r\nUser: WIN-OUNNPISDHIG\\Administrator\r\nLogonGuid: {00000000-568E-5453-0000-0020 ↩
    D5ED0400}\r\nLogonId: 0x4edd5\r\nTerminalSessionId: 2\r\nIntegrityLevel: High\r\ ↩
    nHashType: SHA1\r\nHash: 1DCE4B0F24C40473CE7B2C57EB4F7E9E3E14BF94\r\nParentProcessGuid: ↩
     {00000000-3862-553E-0000-001088D30527}\r\nParentProcessId: 26544\r\nParentImage: C:\\ ↩
    msys\\1.0\\bin\\sh.exe\r\nParentCommandLine: C:\\msys\\1.0\\bin\\sh.exe",
 "Opcode":"Info",
 "UtcTime":"2015.04.27. 13:23",
 "ProcessGuid":"{00000000-3862-553E-0000-001051D40527}",
 "Image":"c:\\Program Files (x86)\\nxlog\\nxlog.exe",
 "CommandLine":"\"c:\\Program Files (x86)\\nxlog\\nxlog.exe\" -f",
 "User":"WIN-OUNNPISDHIG\\Administrator",
 "LogonGuid":"{00000000-568E-5453-0000-0020D5ED0400}",
 "LogonId":"0x4edd5",
 "TerminalSessionId":"2",
 "IntegrityLevel":"High",
 "HashType":"SHA1",
 "Hash":"1DCE4B0F24C40473CE7B2C57EB4F7E9E3E14BF94",
 "ParentProcessGuid":"{00000000-3862-553E-0000-001088D30527}",
 "ParentProcessId":"26544",
 "ParentImage":"C:\\msys\\1.0\\bin\\sh.exe",
 "ParentCommandLine":"C:\\msys\\1.0\\bin\\sh.exe",
 "EventReceivedTime":"2015-04-27 15:23:47",
 "SourceModuleName":"in",
 "SourceModuleType":"im_msvistalog"
}
```


Generally we will want to send the security audit events to a remote server. The following NXLog configuration does that. The important eventlog sources are read by the *im_msvistalog* module. On the output side the event are converted to JSON and then the JSON is inserted into the message part of a syslog record which is sent over a TCP connection to the server.

```
define ROOT C:\Program Files (x86)\nxlog

Moduledir %ROOT%\modules
CacheDir %ROOT%\data
Pidfile %ROOT%\data\nxlog.pid
SpoolDir %ROOT%\data
LogFile %ROOT%\data\nxlog.log
LogLevel INFO

<Extension _json>
    Module xm_json
</Extension>

<Extension _syslog>
    Module xm_syslog
</Extension>

<Input in>
    Module im_msvistalog
    Query <QueryList> \
          <Query Id="0"> \
           <Select Path="Microsoft-Windows-Sysmon/Operational">*</Select> \
           <Select Path="Application">*</Select> \
           <Select Path="System">*</Select> \
```

```
            <Select Path="Security">*</Select> \
          </Query> \
        </QueryList>
</Input>

<Output out>
    Module      om_tcp
    Host        10.0.0.1
    Port        1514
    Exec        to_json(); $Message = $raw_event; to_syslog_bsd();
</Output>

<Route 66>
    Path        in => out
</Route>
```

There are many different options for sending the data, e.g. using NXLog's Binary data format to preserve structured data, using TLS for security and encryption, sending only selected events, etc. We won't be discussing these here, except for the filtering options.

We may not want to store or send all the eventlog records to the server so we need to filter them. There are three different options for filtering Sysmon generated data. The second two apply to any other eventlog record as well.

Filtering data using the Sysmon configuration The most efficient method is to instruct Sysmon not to collect and store the event in the eventlog at all. This can be done with the filtering tags in the Sysmon XML configuration file as discussed earlier.

Using the Query directive The *im_msvistalog* has a Query directive which can be used to specify an XML Query that gets passed to the Windows EventLog API in order to read only the selected events. The Windows Event Viewer can help construct such XML queries. The following example will only collect only process creation event records from the Sysmon source.

```
Query <QueryList> \
        <Query Id="0">\
          <Select Path="Microsoft-Windows-Sysmon/Operational">*[System[(EventID='1') ↩
              ]]</Select>\
        </Query>\
      </QueryList>
```

The event records filtered with the Query directive do not reach NXLog so this might be slightly more efficient than the next native NXLog filtering method.

Filtering with NXLog's log processing language The NXLog log processing language is available for use by all modules and may be easier to write than the XML query syntax provided by the Windows EventLog API that the *im_msvistalog* exposes. The following NXLog style filter statement achieves the same as the XML Query above.

```
Exec if not ($Channel == 'Microsoft-Windows-Sysmon' and $EventID == 1) drop();
```

The following filtering rule will remove event records that are HTTP network connections to a specific server:

```
Exec if $SourceName == 'Microsoft-Windows-Sysmon' and $DestinationPort == 80 and  ↩
    $DestinationIp == 10.0.0.1 drop();
```

We hope that this short tutorial helped to get started with the basics of collecting Windows Security Audit logs. Feel free to drop us a line if you have any questions and comments or are interested in the enterprise offering and support.