Investigators can examine Domain Name Service (DNS) queries to find potentially compromised hosts by searching for queries that are unusual or to known malicious domains. Once the investigator identifies the compromised host, they must then locate the process that is generating the DNS queries. The problem is that Windows hosts do not log DNS client transactions by default, and there is little documentation on the structure of those logs. This paper examines how to configure several modern versions of Windows to log DNS client transactions to determine the originating process for any given DNS query. These configurations will allow investigators to determine not only what host is compromised, but what the malicious process is more quickly.
1. Introduction

Domain Name Service (DNS) makes the internet relatively easy to use for humans by allowing users to request a friendly name (such as tindrasgrove.com) and translating that into a computer-friendly Internet Protocol (IP) address (such as 146.66.99.37). DNS makes the internet easier to use not only for humans, but also for malware.

Malicious actors race against network defenders to evade detection. While the defenders are building signatures to alert on and protect against the actors’ activities, the actors are modifying their behavior to avoid those signatures. Some signatures are more fragile than others, and actors modify some behaviors more easily than others. Bianco (2013) introduced the Pyramid of Pain to explain this concept. According to the Pyramid, while domain names are simple to change, they are more difficult to change than IP addresses. Where possible, this also means that domain names make a more resilient signature compared to IP addresses.

For example, a piece of malware may beacon out to hard-coded servers. If it uses IP addresses, and those servers get taken down, then the malware is no longer able to communicate successfully. If the malware uses a domain name, and a server gets taken down, the malicious actors can easily point the domain to a new server with a new IP address, and the malware will continue to communicate.

In addition to being more resilient, using domain names makes malware look more like normal network traffic. It is extremely rare for users to browse directly to the IP address for a website; there is almost always a DNS query first (Murdoch, 2018, p. 79). By using domain names instead of IP addresses, malware does a better job of blending in and evading detection.

Due to the reliance of most network traffic on DNS, network defenders use it to detect malicious activity. They not only build signatures based on known-bad domains, or known techniques of abusing DNS queries, but can perform statistical analysis on queried domains to determine what is unusual and what is worth investigating further. Defenders use long tail analysis to identify what is uncommon within the environment (Murdoch, 2018, p. 148); queries for domains that are not requested by any other hosts is a common indicator of compromise.

Amanda Draeger, amanda@tindrasgrove.com
Organizations usually monitor DNS requests at the local DNS server or using network monitoring tools such as Zeek or Suricata to capture and log DNS traffic at the network perimeter. This method provides a high level of efficiency as the organization only needs to monitor one location instead of all the local traffic. Unfortunately, this leaves the organization blind to what their corporate systems are doing when they are off the corporate network, such as when employees travel with their laptops.

Monitoring DNS queries at the network perimeter provides an excellent means of identifying compromised hosts by IP address and/or hostname. As the investigation moves to the host, the analyst must determine what process initiated the DNS request. Windows hosts, which are the most common for enterprise users, do not log DNS client transactions by default. Thus, for many analysts, the only method of determining what process initiated a DNS query is to closely watch the system and hope to catch the process making the same request again. Such live analysis is not always possible or desirable, depending on the organization's incident response policies.

Analysts need to be able to pivot from network evidence to host evidence more efficiently—that means logging DNS queries on the client. More than just logging the requests, the logs need to identify the process that originated the DNS query. This requires configuration changes to host systems to capture and correlate this information.

2. Research Method

2.1. Lab Design

A virtualized lab environment was used to simultaneously run the system generating DNS requests (Windows host) and a network monitor. A PFSense firewall provided the connection to the internet for the Windows host and acted as the network gateway, as well as the local DNS and Dynamic Host Configuration Protocol (DHCP) server. The internal interface of the firewall was assigned an IP address of 172.16.2.1.

A Security Onion (SO) virtual machine (VM) acted as the network monitor. The Zeek software package on SO provided logging of all DNS requests, as well as their responses, within the virtual network. Traffic was copied to the network monitor by configuring SO with a promiscuous port, allowing it to read all traffic on the local network, including traffic that was not addressed to it. This setup was used to provide the

Amanda Draeger, amanda@tindrasgrove.com
same data as the enterprise network practice of logging DNS requests at the network perimeter.

The Windows hosts were VMs provided by Microsoft for test and development purposes. Windows Server 2019, Windows Server 2016, and Windows Server 2012 R2 (Microsoft, Microsoft Evaluation Center, 2019) were used to test server environments. Windows 10, Windows 8.1, and Windows 7 (Microsoft, Free Virtual Machines from IE8 to MS Edge, 2019) represented user operating systems. Each system was fully updated, had the Firefox web browser installed, and had the time zone set to Coordinated Universal Time (UTC). While six different versions of Windows were tested, only one Windows host was on the network at any time. The different VMs came with different versions of Internet Explorer and Edge installed, so Firefox provided a consistent baseline across all versions of Windows.

2.2. Test Design

All tests were conducted with the SO VM monitoring and logging all traffic from the Windows host. This allowed for correlation of host artifacts with network artifacts. Additionally, as the network artifacts included the source port for the DNS request, the network traffic provided a starting point for identifying what process initiated the request.
Each Windows host had its DNS Client Events Operational log enabled. The Firefox web browser was used to generate traffic to www.tindrasgrove.com. Evidence of that DNS query was found within the DNS Client logs. The process identifier (PID) that created the DNS query was identified. If the PID matched the PID for the running Firefox process, the results were documented. If the PID did not match the Firefox PID, additional configuration changes were made until the DNS query could be traced to Firefox. These configuration changes included additional log collection and modification of running services.

3. Findings and Discussion

3.1. Windows Event Logs

Windows Vista made some significant changes from previous versions of Windows, including how Windows handles event logs. In addition to significant performance improvements, the format was changed to be Extensible Markup Language (XML)-based (Lee, 2018, p. 273). This allowed for more verbose logs that are more easily digestible. The combination of this change in format with the integration of PowerShell makes parsing logs much easier than the previous binary log format.

One of the classic log types is the Security log. While it does not contain useful DNS information, it provides a baseline for learning log structure and methods for reading event logs. For example, a single line of PowerShell can return the first event from the Security log, as shown in Figure 2.

![Figure 2: Event log using FilterHashtable](image-url)
While useful, this does not provide the full scope of information that is available in the new log formats. A hash table is a mapping of keys to values. In this example, the ID key has a value of 5061. This allows filtering based on key-value pairs. The Format-List * command, used in Figure 3, allows listing of the entire log entry, including all possible keys.

For the hash table’s purposes, the Message key is a single field that cannot be further filtered. One may search for a substring within the Message field but may not specify that string is only within the Algorithm Name. To achieve that level of specificity when searching through logs, one must look at the underlying XML.
The System section contains many of the fields that were in the hash table, while the EventData section contains the fields that populated the Message section. Each event provider and EventID will have a unique format. To query based on these fields, PowerShell allows use of XML and XPath. It is now possible to select only those Security logs where the Algorithm Name is RSA.

Figure 4: XML of an event log

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  - <System>
    <Provider Name="Microsoft-Windows-Security-Auditing" Guid="{54B49625-5478-4994-A58A-3E380328C30D}" />
    <EventID>5061</EventID>
    <Version>0</Version>
    <Level>0</Level>
    <Task>12290</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8000000000000000</Keywords>
    <TimeCreated SystemTime="2019-03-10T14:00:37.739736600Z" />
    <EventRecordID>13301</EventRecordID>
    <Correlation ActivityID="{368A56A-D743-0000-1A57-8A3643D7D401}" />
    <Execution ProcessID="508" ThreadID="2636" />
    <Channel>Security</Channel>
    <Computer>WIN-7LP09KICDC</Computer>
    <Security />
  </System>
  <EventData>
    <Data Name="SubjectUserSid">S-1-5-18</Data>
    <Data Name="SubjectUserName">WIN-7LP09KICDC</Data>
    <Data Name="SubjectDomainName">WORKGROUP</Data>
    <Data Name="SubjectLogonId">0x3e7</Data>
    <Data Name="ProviderName">Microsoft-Software Key Storage Provider</Data>
    <Data Name="AlgorithmName">RSA</Data>
    <Data Name="KeyName">5cc3f1ea-2e44-68e3-d143-c3bfe41e90c</Data>
    <Data Name="KeyType">% 2500</Data>
    <Data Name="Operation">% 2480</Data>
    <Data Name="ReturnCode">0x0</Data>
  </EventData>
</Event>
```

Figure 5: Security log using XML filtering

```
PS C:\Users\Administrator> Get-EventLog -FilterXML "<querylist><query><Query><Select Path='Security'>[[EventData[Data[@Name='AlgorithmName']=] and (Data[Data[@Name='AlgorithmName']=])]]</Select>" -QueryList
```

SelectPath = 'Security' tells the command to only look in the Security logs. EventData narrows the query down to the bottom half of the XML. Data[@Name = 'AlgorithmName'] identifies which Data field is being queried. Finally, and (Data = 'RSA') is the value being searched for.
3.2. DNS Client Events Log

Windows Vista not only introduced a new event log format, but new event log types. In addition to the three classic logs (Application, System, and Security) there are now many log types under *Applications and Services Logs*. This provides much more detailed logging and allows applications to create unique logs. The *DNS Client Event – Operational* log is one of these new log types.

On all tested versions of Windows, the DNS Client log was disabled by default. Enabling the log consisted of browsing to the log in Event Viewer and selecting the option to enable the log. Events immediately began populating as Windows is constantly looking to communicate with other systems.

Many different types of DNS events occurred. These ranged from events recording timeouts to events recording the DNS queries and responses, in both the local cache and network requests. For a full listing of event IDs observed, refer to Appendix A. The most useful Event ID for investigations into DNS client behavior is 3006. This log captures useful information regardless of whether the query is sent to the local cache or is sent over the network to a DNS server. Other DNS event types, such as 3009 and 3016, are specific to local cache or network queries.

In the below DNS log, there are three fields of primary interest, highlighted in green boxes. The first is *EventID*. As mentioned, restricting analysis to Event ID 3006 will yield the most useful data for identifying the originating process for a given DNS query. The second field is *Execution ProcessID*, which records the PID that asked the DNS client to make a DNS query. The third field is *QueryName*, which records the domain queried.
The goal is to look through all DNS queries (Event ID 3006), search for a domain name of interest (tindrasgrove.com) and return the PID that made the query. This is most easily achieved using the following command:

```
Get-WinEvent -FilterHashtable @{'logname='Microsoft-Windows-DNS-Client/Operational' ; id=3006'} | Where-Object {$_.'Message -like '*tindrasgrove*' } | Format-Table -property TimeCreated, ProcessID, Message
```

This command will search the DNS Client Operational log for all events with ID 3006, where the message key contains the string “tindrasgrove” and will output those events into a table that displays the *TimeCreated, ProcessID, and Message* fields. For this query, *FilterHashtable* is preferable to *FilterXML* because the implementation of XPath

Amanda Draeger, amanda@tindrasgrove.com
used by Windows Event Log places restrictions on the expressions used (Microsoft, 2018). This means that wildcard operators cannot be used, which can be troublesome when trying to query a base domain, but the logs contain a subdomain. Not being able to use wildcard operators in this scenario means that there is no means of searching for both www.tindrasgrove.com and mail.tindrasgrove.com with a single query.

In the logs (below), a single visit in a web browser to tindrasgrove.com resulted in four separate 3006 log entries. This is due to the different query options used (namely querying both the local cache and the network, and both IPv4 and IPv6). The logs also reflect queries for both the domain itself and the www subdomain. In all four of the most recent logs, the calling PID for the DNS query is 4072. The next step of the investigation is to see what program that PID belonged to at the time of the DNS query.

3.3. Windows Process Creation

The highly constrained environment of conducting a test in a lab makes it practical to correlate the PID from the event log against currently running processes. In an incident response scenario this is not the case. Process IDs need to be captured and logged for retroactive analysis. Sysmon (Microsoft, 2017), part of the Sysinternals suite from Microsoft, is a program that effectively logs process creation.

Sysmon runs as a system service to generate information regarding the behavior of the system. It logs the information it generates to the Windows event log, in the Sysmon – Operational log. This allows the analyst to use the same methodology to query Sysmon events as was used to query the DNS Client logs. In order to define which activities get monitored, Sysmon uses an XML configuration file. SwiftOnSecurity has published a starting point for this configuration file at their GitHub (SwiftOnSecurity, 2018). This configuration file was used without modification in the lab environment.
The Sysmon logs are structured similarly to all other Windows event logs. The main difference is that they are highly customizable depending on the configuration file. Below is the Event ID 1: Process Creation log that corresponds to the DNS query above.

```
- <Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  - <System>
    <Provider Name="Microsoft-Windows-Sysmon" Guid="{5770385F-C22A-43E0-BF4C-06F5698FFBD9}"/>
    <EventID>1</EventID>
    <Version>5</Version>
    <Level>4</Level>
    <Task>1</Task>
    <Opcode>0</Opcode>
    <Keywords>0x8000000000000000</Keywords>
    <TimeCreated SystemTime="2019-03-10T19:50:25.512357000Z"/>
    <EventRecordID>2009</EventRecordID>
    <Correlation />
    <Execution ProcessID="1488" ThreadID="2112"/>
    <Channel>Microsoft-Windows-Sysmon/Operational</Channel>
    <Computer>WIN-7LPK09KICD</Computer>
    <Security UserID="S-1-5-18"/>
  </System>
  - <EventData>
    <Data Name="RuleName"/>
    <Data Name="UtcTime">2019-03-10 19:50:25.482</Data>
    <Data Name="ProcessGuid">{3A504E03-6A81-5C85-0000-0010A1365200}</Data>
    <Data Name="ProcessId">4072</Data>
    <Data Name="CommandLine">C:\Program Files (x86)\Mozilla Firefox\firefox.exe</Data>
    <Data Name="FileVersion">65.0.1</Data>
    <Data Name="Description">Firefox</Data>
    <Data Name="Product">Firefox</Data>
    <Data Name="Computer">Empresa: Mozilla Corporation</Data>
  </EventData>
</Event>
```

Figure 8: XML of Sysmon Event ID 1: Process Creation

For this log, the notable fields are in the Event Data section. The ProcessID is the field to query based on the PID returned from the DNS Client logs. CommandLine will report the program and any command-line switches that were used to start the process. This should suffice to verify if the DNS Client logs are reporting the true calling process of the DNS query. This can be achieved by running the following command:

```
```

Amanda Draeger, amanda@tindrasgrove.com
Get-WinEvent -FilterXML "<QueryList><Query><Select Path='Microsoft-Windows-Sysmon/Operational'>*[System[EventID=1]] and *[EventData[Data[@Name='ProcessID']]="4072']]></Select></Query></QueryList>" | Format-List TimeCreated, Message

FilterXML works for this command because the information searched for is a number; there is no wildcard required unlike with the domain name described above. This works particularly well because parsing the XML allows searching for that number in a precise field, instead of reading the message field as a single field. For this command, the search is restricted to Sysmon Event ID 1 (Process Creation) logs. From there, the desired PID (4072) is searched for in the ProcessID data field. It is also possible to search all of the Name fields, but this will return too many logs, as the number may show up in multiple fields, such as the ParentProcessID.

![Figure 9: Results of query for process ID](image)

Based on the output of the command, and a rough correlation of timestamps, it is evident that the process that generated the DNS request for tindrasgrove.com (PID 4072) is Firefox. This was the program used to visit the website, so the DNS Client logs do, in fact, record the true process that generated the DNS queries. There is no broker process at work. This was the case in Windows 8.1, Windows 10, Server 2012R2, Server 2016, and Server 2019. A script that can be used to perform these steps is in Appendix B.
3.4. Windows 7

Windows 7 is an outlier in terms of the behavior of current versions of Windows. While it does have the significantly-enhanced logging of Windows Vista and beyond, there were changes made between Windows 7 and 8. As one example of these changes, the Event IDs in the DNS Client logs in Windows 7 are completely different from 8.1 and 10. There are also not as many events that are logged by default when enabling the logs. In this example, only two events were created by a visit to tindrasgrove.com, and neither of them is Event ID 3006, which was the most useful ID in Windows 10.

Another significant change is the way that Windows 7 interprets the originating process for the DNS request. The logged ProcessID, by default, is an instance of svchost, not the web browser used to visit the page. This means that there is more work that needs to be done before an analyst can determine what process generated the DNS request.

By default, svchost is acting as a broker service. It is passing DNS requests to a different Windows service that performs local DNS caching that is confusingly named the DNS Client service. Disabling this service does not prevent the system from handling DNS requests; it primarily handles local caching and name registration. Although there is nothing in the service’s description that suggests that this is the case, disabling this service has the side effect of changing what the DNS Client logs report as the originating process from svchost to the actual originating process. Once the service has been disabled, the same PowerShell commands allow retrieval of the relevant logs.

Amanda Draeger, amanda@tindrasgrove.com
While this did not cause any adverse effects in a controlled lab environment with a single client that is not joined to a domain, disabling the DNS Service should not be done in a production environment without testing. In his article, J. Werner suggests that disabling the DNS Service is the cause of some programs failing to connect to their update servers, leaving the system in a vulnerable state (2017).

As of the time of this writing, Windows 7 is less than one year from reaching its end of support (Microsoft). This fact, combined with the differences in log format and process logging, means that organizations should work to move as many systems as can be migrated to supported operating systems. The logging differences add complications because different configurations signatures are required to get the same information. In a heterogeneous environment, this means having to do twice the work. Also, if any of the systems have the DNS Client service enabled, then they will lack valuable information.

4. Recommendations and Implications

The linking of DNS query to the originating process is achievable using only tools that are built into Windows (event logs, PowerShell) or distributed by Microsoft (Sysmon). Despite this being a native capability, the administrator needs to take care when making any changes to their systems.

Amanda Draeger, amanda@tindrasgrove.com
4.1. Recommendations for Practice

Whenever an organization is considering making changes to which items are being logged, it needs to carefully consider the log rotation interval, as well as overall log volume. To plan its logging strategy, the organization should consider their ideal, as well as their minimum viable, retention period. This should inform how the logs are tuned, and where they are stored.

DNS Client logs are capable of being extremely noisy as there are many types of logs, with a single DNS request potentially generating a dozen different log entries. Windows is also a chatty operating system, and it is constantly looking for other systems. The default log size (1028 KB) is easily overwritten in several hours with normal computer use. In order to minimize this background noise, filtering down to only Event ID 3006 will ensure that the most important information (the requesting process ID) is captured. This will decrease the number of logs generated, which will extend the amount of time that logs can be retained before they rotate.

The Sysmon configuration provided by SwiftOnSecurity was thoroughly tested in their environment, but it should be tested and tuned by any organization that adopts it. Process creation logs add up quickly, and these are not the only logs that Sysmon can generate. If an organization is using Sysmon for process creation logging only, then it should be configured to only log those events to reduce log volume.

Once an organization has decided what data to log and how long it should be kept, they need to decide where those logs will be stored. The three choices are to keep the logs stored locally on the host that generated them, to ship them to a centralized log storage server, or to ingest them into a Security Information and Event Management (SIEM) device. In order to make that decision, the organization must consider how they plan to use the logs. If they have signatures that will alert, then a SIEM is appropriate. If they want to be able to search through the entire enterprise’s worth of logs, then centralized log management will work. If, however, they only want the logs available in case of an incident where a specific host has been compromised, then leaving them on the host may be the most appropriate option. In this case the maximum allowable log size must be adjusted to allow a useful retention period.

Amanda Draeger, amanda@tindrasgrove.com
The primary use case for this research project was incident response, so there may not be a compelling reason to centralize DNS Client logs. However, different organizations may find that they have additional use cases that make log aggregation worth the additional volume.

### 4.2. Implications for Future Research

Another interesting use case for logging DNS queries at the client, not just the perimeter, is that not all clients spend all their time on a network with a closely-monitored perimeter. Teleworkers and road warriors are some classic cases of computers that spend time off-premises. The shift towards perimeterless networks in some organizations also makes monitoring at the network boundary difficult. Small office and home office (SOHO) users also find themselves without centralized logging, so logging at the client can be the only monitoring available.

An interesting avenue of future research may be to use the local DNS Client logs to create meaningful detections. Slicing out the visited domains and comparing them against a blacklist or a reputation service to identify known malicious activity is one potential use. Another is to perform long-tail analysis to find rare domains; what is unusual is often malicious. Generally speaking, any detections that are useful at the perimeter should be useful on the client; the tools just need to be built to perform that analysis on the client.

While this research has focused on the Windows DNS Client logs, there are many more events that Windows is capable of logging. Future researchers may be inspired to examine some of those logs more closely to find more techniques for detecting malicious activity by using the built-in tools that Microsoft has provided.

### 5. Conclusion

Incident responders face many challenges, not least of which is dealing with the pivot from a network-based investigation to a host-based investigation. That transition can be made much easier by collecting data on the host, such as DNS queries, that link the processes running on the host with the traffic observed on the network. Fortunately, this can be done using only the tools that Microsoft has provided: event logs, Sysmon, and PowerShell. As with all changes to a system’s configuration, administrators need to

Amanda Draeger, amanda@tindrasgrove.com
define their log retention requirements and tune logs to the minimum volume that
provides useful information in order to meet those requirements. With some careful
planning and implementation of a few configuration changes, incident responders can
have the data they need to do their jobs far more efficiently and obtain a more complete
picture of the activity on their networks.
References


Amanda Draeger, amanda@tindrasgrove.com
## Appendix A
### DNS Client Log Event IDs (Windows 8+/Server 2012+)

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>There are currently no IPv4 DNS servers configured for any interface on this host. Please configure DNS server settings, or renew your dynamic IP settings.</td>
</tr>
<tr>
<td>1001</td>
<td>Interface: Ethernet Total DNS Server Count: &lt;#&gt; Index: &lt;#&gt; Address: &lt;address&gt;</td>
</tr>
<tr>
<td>1013</td>
<td>Name resolution for &lt;domain&gt; timed out after none of the configured DNS servers responded.</td>
</tr>
<tr>
<td>1015</td>
<td>Name resolution for the name &lt;name&gt; timed out after the DNS server <a href="">address:port</a> did not respond.</td>
</tr>
<tr>
<td>1016</td>
<td>A name not found error was returned for the name &lt;name&gt;. Check to ensure that the name is correct. The response was sent by the server at &lt;ip&gt;</td>
</tr>
<tr>
<td>1017</td>
<td>The DNS server's response to a query for the name &lt;name&gt; indicates that no records of the type queried are available, but could indicate that other records for the same name are present.</td>
</tr>
<tr>
<td>1019</td>
<td>There are currently no IPv6 DNS servers configured for any interface on this host. Please configure DNS server settings, or renew your dynamic IP settings.</td>
</tr>
<tr>
<td>1022</td>
<td>Name resolution for the name, &lt;name&gt;, will not fall back to LLMNR or NetBIOS</td>
</tr>
<tr>
<td>3006</td>
<td>DNS query is called for the name &lt;name&gt;, type &lt;type&gt; query options &lt;option&gt;, Server List, isNetwork query &lt;#&gt;, network index &lt;#&gt;,</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>3007</td>
<td>DnsQueryEx for the name <code>&lt;name&gt;</code> is pending</td>
</tr>
<tr>
<td>3008</td>
<td>DNS query is completed for the name <code>&lt;name&gt;</code>, type <code>&lt;type&gt;</code>, query options <code>&lt;options&gt;</code> with status <code>&lt;status&gt;</code> Results <code>&lt;results&gt;</code></td>
</tr>
<tr>
<td>3009</td>
<td>Network query initiated for the name <code>&lt;name&gt;</code> in network index <code>&lt;index&gt;</code> with interface count <code>&lt;#&gt;</code> with first interface name <code>&lt;name&gt;</code>, local address <code>&lt;address&gt;</code> and DNS Servers <code>&lt;servers&gt;</code></td>
</tr>
<tr>
<td>3010</td>
<td>DNS Query sent to DNS Server <code>&lt;address&gt;</code> for name <code>&lt;name&gt;</code> and type <code>&lt;type&gt;</code></td>
</tr>
<tr>
<td>3011</td>
<td>Received response from DNS Server <code>&lt;server&gt;</code> for name <code>&lt;name&gt;</code> and type <code>&lt;type&gt;</code> with response status <code>&lt;status&gt;</code></td>
</tr>
<tr>
<td>3016</td>
<td>Cache lookup called for <code>&lt;name&gt;</code>, type <code>&lt;type&gt;</code>, options <code>&lt;options&gt;</code> and interface index <code>&lt;#&gt;</code></td>
</tr>
<tr>
<td>3018</td>
<td>Cache lookup for name <code>&lt;name&gt;</code>, type <code>&lt;type&gt;</code>, and option <code>&lt;options&gt;</code> returned <code>&lt;status&gt;</code> with results <code>&lt;results&gt;</code></td>
</tr>
<tr>
<td>3019</td>
<td>Query wire called for name <code>&lt;name&gt;</code>, type <code>&lt;type&gt;</code> interface index <code>&lt;index&gt;</code> and network index <code>&lt;index&gt;</code></td>
</tr>
<tr>
<td>3020</td>
<td>Query response for name <code>&lt;name&gt;</code>, type <code>&lt;type&gt;</code> interface index <code>&lt;index&gt;</code> returned <code>&lt;status&gt;</code> with results <code>&lt;results&gt;</code></td>
</tr>
</tbody>
</table>
Appendix B
DNS Log Query Script (Windows 8+/Server 2012+)

$domain = Read-Host -Prompt 'Domain to query'

$print1 = Get-WinEvent -FilterHashtable @{logname='Microsoft-Windows-DNS-Client/Operational'; id=3006} | Where-Object {$_._Message -like "*\$domain*"} | Format-Table -property TimeCreated, ProcessID, Message
$print1

$process = Read-Host -Prompt 'Process to query'

Get-WinEvent -FilterXML "<QueryList><Query><Select Path='Microsoft-Windows-Sysmon/Operational'>*[System[EventID=1]] and *[EventData[Data[@Name='ProcessID']='$process']]]</Select></Query> </QueryList>" | Format-List TimeCreated, Message

Amanda Draeger, amanda@tindrasgrove.com
<table>
<thead>
<tr>
<th>Training Event</th>
<th>Location</th>
<th>Dates</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS Amsterdam August 2020 Part 1</td>
<td>Amsterdam, NL</td>
<td>Aug 03, 2020 - Aug 08, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS FOR508 Canberra August 2020</td>
<td>Canberra, AU</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Amsterdam August 2020 Part 2</td>
<td>Amsterdam, NL</td>
<td>Aug 17, 2020 - Aug 22, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Virginia Beach 2020</td>
<td>Virginia Beach, VAUS</td>
<td>Aug 31, 2020 - Sep 05, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Philippines 2020</td>
<td>Manila, PH</td>
<td>Sep 07, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS London September 2020</td>
<td>London, GB</td>
<td>Sep 07, 2020 - Sep 12, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Baltimore Fall 2020</td>
<td>Baltimore, MDUS</td>
<td>Sep 08, 2020 - Sep 13, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>Threat Hunting &amp; Incident Response Summit &amp; Training 2020</td>
<td>New Orleans, LAUS</td>
<td>Sep 10, 2020 - Sep 17, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS Munich September 2020</td>
<td>Munich, DE</td>
<td>Sep 14, 2020 - Sep 19, 2020</td>
<td>Live Event</td>
</tr>
<tr>
<td>SANS OnDemand</td>
<td>OnlineUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>