Global Information Assurance Certification Paper

Detecting Security Incidents Using Windows Workstation Event Logs

GIAC (GCIH) Gold Certification

Author: Russ Anthony, ranthony00@gmail.com
Advisor: Robert Vandenbrink
Accepted: June 19th 2013

Abstract

Windows event logs can be an extremely valuable resource to detect security incidents. While many companies collect logs from security devices and critical servers to comply with regulatory requirements, few collect them from their windows workstations; even fewer proactively analyze these logs. Collecting and analyzing workstation logs is critical because it is increasingly at the workstation level where the initial compromise is happening. If we are to get better at detecting these initial compromises then it is imperative that we develop an efficient, common sense approach to collecting and analyzing these events.

1. Introduction

Windows event logs are a critical resource when investigating a security incident and aide in the determination of whether or not a system has been compromised. Event logs are an integral part of constructing a timeline of what has occurred on a system. While many companies collect logs from security devices and critical servers to comply with regulatory requirements such as FISMA, HIPAA and PCI-DSS, few collect events from Windows workstations and even fewer proactively analyze workstation logs.

The Advanced Persistent Threat (APT) attack life cycle as defined by Mandiant (Mcwhorter, 2013) consists of the following steps: the initial compromise, establishing a foothold, escalating privileges, internal reconnaissance, moving laterally, maintaining presence, and completing the mission. Because it is at the workstation level that most initial compromises happen, and additionally where many of the subsequent steps of the attack life cycle also happen; it follows that looking at workstation events is a very logical place to focus attention when trying to combat APT style attacks. Collecting logs from workstations however, does present some challenges. First, workstations can produce huge volumes of event logs. It is therefore important that this vast volume of logs be collected efficiently. Secondly, there is no integrated mechanism in Windows to transport the logs to a centralized collector. Finally, how does a security professional analyze this huge volume of logs to discern targeted intrusions?

The solution is what Gartner (Chuvakin, 2012) refers to as an output-driven collection and analysis strategy. An output-driven strategy means you only collect any particular event if you know beforehand what you are going to do with that event and only if that event is going to build towards some known output such as an alert or a report metric. Put differently, high quality events ***in*** will produce high quality alerts ***out***. This strategy allows for prioritization right from the start. It allows you to decide what information is to be obtained from log collection and analysis. Done properly, it can be an inexpensive, efficient, and valuable addition to the overall security monitoring process. The downside to the output –driven strategy is that security practitioners must know precisely what to look for before beginning. This means a lot of thought and upfront planning is required in the planning stages.

Russ Anthony, ranthony00@gmail.com

The output-driven strategy is the polar opposite of most typical Security Information and Event Monitoring (SIEM) deployments which Gartner (Chuvakin, 2012) refers to as an input-driven strategy. An input-driven strategy calls for collecting "everything" with the hope that someday you will figure out what to do with it.

The other piece of this solution is to use the syslog protocol for efficient transmission of the Windows events over the network to a centralized collector or SIEM. The Syslog protocol as defined by RFC 5424 (Gerhards, 2009) utilizes a layered architecture to efficiently transmit and collect event messages. The layered architecture consists of a local collector or an agent, a network transport mechanism (UDP or TCP) and a centralized collector. In order to collect Windows event logs using syslog, a third party client is required, as Windows does not natively support the syslog protocol. There are a few things to consider when selecting a third party agent. First, the ability to accurately filter events is a must have for an output-driven collection and analysis strategy. Secondly, the ability to manage the filters from an enterprise level is very important because it provides the ability to quickly change filters as the threat landscape changes. Finally, if workstations ***live*** off network during travel for example, special considerations should be made to implement an agent that will buffer the events until the workstations return to the network. For those that have a limited budget or simply prefer an open source solution, Appendix 2 details a solution using ***eventlog-to-syslog*** as the windows syslog agent, ***rsyslog*** as the centralized collector, and ***Simple Event Correlator (SEC)*** for the analysis and alerting.

2. Filtering the Event Logs

2.1 Event Filtering - The First Cut.

Per the output-driven strategy, security professionals only want to collect those events where there is a clear understanding of how it will be later used. The most effective way to accomplish this is by creating a whitelist filter. A whitelist is a list of all the events to be collected, everything else that does not match the whitelist is discarded. This whitelist should be applied at the client level to minimize the number of events, which must be forwarded to the collector and to minimize the number of events that have

to be analyzed. This whitelist does not change in any way what events are stored on the local workstation. It only means the events that do not match are not forwarded to the collector.

Table 1 lists a series of regular expressions that will match interesting Windows 7 events that will later be used in the analysis. A similar list for Windows XP Events is located in the Appendix. The Windows Event ID numbers can also be used to create the white list though in some case they are not as effective as the regular expressions. Two excellent resources for looking up Windows events and their associated Event ID numbers are www.ultimatewindowssecurity.com and www.eventid.net.

Windows 7 regular expressions	SOURCE	EventID Number
".*APPCRASH.*"	Application	1001
".*he protected system file.*"	Application	64004
".*EMET_DLL Module logged the following event:.*"	Application	2
.*your virus/spyware.*	Application	Varies
".*A new process has been created\..*"	Security	4688
".*A service was installed in the system\..*"	Security	4697
".*A scheduled task was created\..*"	Security	4698
".*Logon Type:[\W]*(3|10).*"	Security	4624, 4625
".*\\Software\\Microsoft\\Windows\\CurrentVersion\\Run.*"	Security	4657
".*service terminated unexpectedly\..*"	System	7034
".*service was successfully sent a.*"	System	7035
".*service entered the.*"	System	7036
".*service was changed from.*"	System	7040

Table1: Windows 7 regular expression white list

2.2 Event Filtering - The Second Cut

The first filter illustrated a whitelist of all the known event types that are desired to be detected. The second filter will be a blacklist. A blacklist is defined as a list of things that are not wanted. The blacklist will be applied in addition to the whitelist. The blacklist will serve to remove any unwanted events (noise) that made it through the first filter. The blacklist can be applied at the collection point or preferably at the client level if the agent supports both whitelist and blacklist filtering. Of all the events that are collected with the white list, the overwhelming majority of them will be "***new process created***" events. The goal is to baseline all of the normal "***new process created***" events in an environment and add them to the black list. There will be some "***noisy***" but interesting "***new process created***" events such as ***svchost.exe*** and ***cmd.exe*** that will not be added to the black list as they are required for the output-driven strategy.

The method to create a list of new processes to populate our black list is quite simple to create. The method consists of collecting a few weeks worth of unfiltered new process created events and then using utilities such as ***grep***, ***sort*** and ***uniq*** to create an ordered list of all the different process names, process paths and the frequency of their use. For example:

```
root@                    logs
[root          logs]# grep 'EventID 4688' win7.log > new_process.log
[root@         logs]# cut -d ":" -f  13-14 new_process.log | sort | uniq -c  > uniq_processes.lo
[root@         logs]# sort -nr   uniq_processes.log >  finished_process_list.log
[root@         logs]# cat finished_process_list.log | more
431142   C:\Windows\System32\SearchProtocolHost.exe
375458   C:\Windows\System32\SearchFilterHost.exe
254458   C:\Windows\System32\SearchIndexer.exe
```

The first command uses ***grep*** to search the file win7.log looking for the string 'EventID 4688' and writes anything that matches to a new file called new_processes.log. The second command uses the command cut to separate each event log on the delimiter ":" and then prints the fields 13 and 14 which is the process name and patch. The ***Sort*** command sorts the list in alphabetic order. The ***uniq*** command makes sure there are no duplicate entries. The last command resorts the list in reverse numeric order to give you output that will show you the most frequently used processes. This will allow you to build a blacklist of common windows processes such as listed in table 2.

Path and Name of process
".*C:\Windows\System32\SearchProtocolHost.exe.*"
".*C:\Windows\System32\SearchFilterHost.exe.*"
".*C:\Windows\System32\SearchIndexer.exe.*"
etc…..

Table 2: Regular Expression Black List.

2.3 Output Driven Filtering Summary

Diagram 2 visually illustrates the effectiveness of the output-driven filtering strategy. For every one million events generated by the workstations, only 300,000 events make it past the first cut and only 90,000 make it past the second cut. In total, this is a 90% reduction in the number of event logs that have to be processed, analyzed and archived. Now that the logs have been filtered to a manageable level, analysis and alert creation can be performed for interesting events.

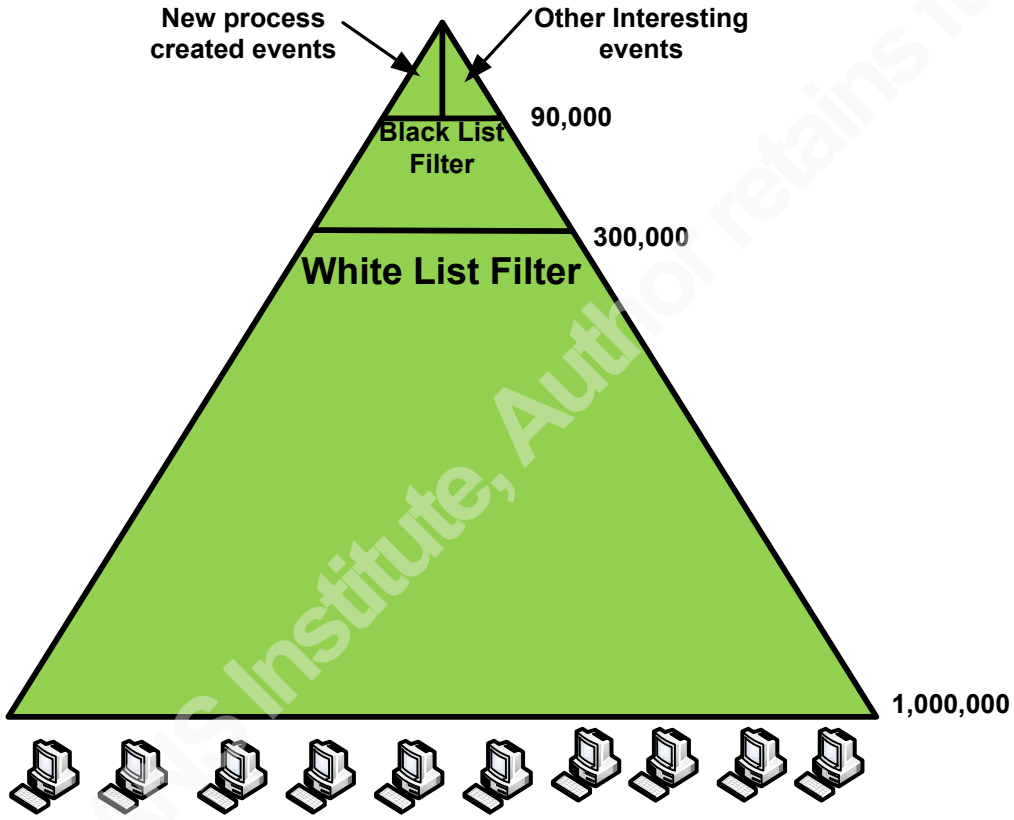


Diagram 1 – output-driven filtering summary

3. Analysis of the Windows Event Logs

3.1 Detecting the Initial Compromise

One way to detect the initial compromise is to detect an unusual running process. Many of the same techniques used to look for malware via memory analysis such as looking for unusual processes names, unusual process paths and unusual process relationships can also be used when analyzing "***new process created***" events (EventID 4688|592).

One type of unusual process name to look for is a common system process name that has been misspelled. Because the spelling is so close, this misspelling will likely go unnoticed even to the trained eye. This is a technique used to hide a malicious process in plain sight. Which of these three processes would you consider to be malicious, scvhost.exe, iexplorer.exe, or svcdost.exe (Torres 2013)? Hopefully, all of them!

Another unusual process name is any process that has a path that begins with a lower case drive letter. A lower case drive letter suggests that the process was started on the command line, or from a script or batch file. This strategy is based on the Volatility plugin known as ***suspicious*** written by Jesse Kornblum for the SANS class FOR 526: Windows Memory Forensics In-Depth.

Also suspicious is any process name that contains a long string of empty spaces. This is a technique used to trick the end user into thinking the file is something it is not. Consider the following process name: "Employee Hanbook.pdf .exe" Note the spaces after the .pdf. This is an example of a malicious executable file that pretends to be a PDF document as shown by Mandiant (Mcwhorter 2013).

Every process consists of a process name and a process path. Any common Windows process that is running from a non standard path should be considered extremely suspicious. This is another technique used to hide malicious processes in plain sight. Fortunately, detecting unusual process paths is easily accomplished with some simple logic.

- Alert on < windows_process_name > unless the path is correct.
- Alert on svchost.exe unless its path is C:\Windows\System32\svchost.exe
- Alert on explorer.exe unless its path is C:\Windows\explorer.exe

The following are two examples of output that such logic would discover which should be considered very suspicious even though they visually may look quite normal.

- C:\Windows\System32\explorer.exe.
- C:\Windows\svchost.exe

Every new process created event contains a process ID number (PID) and a parent process ID number (Parent PID or PPID). The PID is a number that is given to each new process as it is created to uniquely identity that process. The Parent PID represents the process that created the new process. These numbers allow for the analysis of parent /child relationships between processes. "Most system processes have well defined parents; cmd.exe should not be the parent of lsass.exe. Most user processes are started by Explorer.exe. It's suspicious when they're not. Some system processes should never start programs; lsass.exe should not start cmd.exe" (Kornblum, 2010).

Another technique of monitoring ***new process created*** events is to sort them to show which processes are least frequently used. The least frequently used processes are by definition unusual. Unusual events are outliers from common or standard processes and should be considered suspicious. This technique used to detect or search for malware is called the Least Frequency of Occurrence (LFO) principle first described by Pete Silberman (Carvey, 2010). The event logs can be archived in order to build up a long term historical record of all the least frequently used processes. Going forward, if a new malicious process is discovered first hand or through a shared Indicator of Compromise (IOC), the record can be parsed to determine if and when that particular process was ever executed in the environment.

A second way to detect intrusions is to examine suspicious events as described in the SANS Intrusion Discovery Cheat Sheet (SANS, n.d.) for Windows. One example of a suspicious event is when an application crashes (EventID 1001|4097). While there can be many reasons for an application to crash, one more sinister reason is due to a buffer overflow attack. "Buffer overflow errors are characterized by the overwriting of memory fragments of the process, which should have never been modified intentionally or unintentionally. Overwriting values of the IP (Instruction Pointer), BP (Base Pointer) and

other registers causes exceptions, segmentation faults, and other errors to occur" (OWASP, 2009). Crashes of applications such as Adobe Reader, Adobe Acrobat, Adobe Flash, and Microsoft Office documents may be indicative of a ***Spear Phishing*** attack. A spear phishing attack is a very effective technique that is commonly deployed by the APT. The Spear Phish is a well crafted email that contains a malicious attachment such as a weaponized PDF or a hyperlink to a web site that contains malicious code.

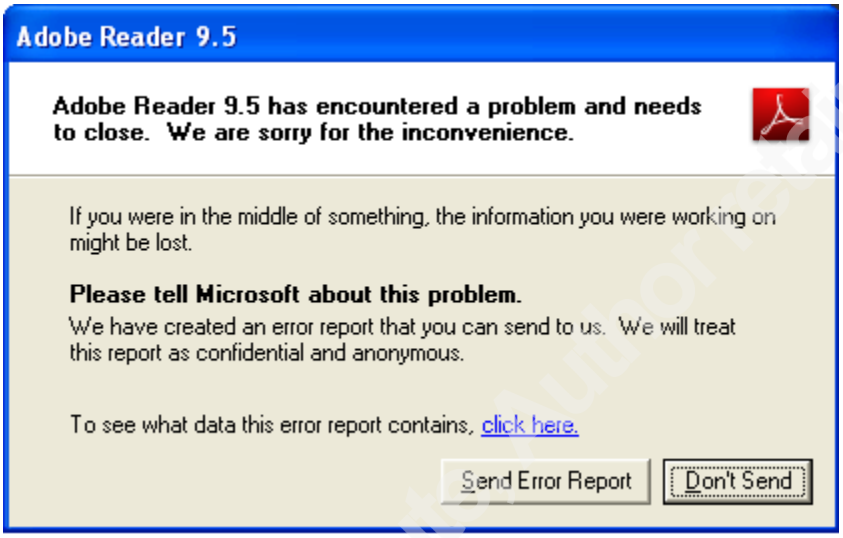


Diagram 2: Adobe Reader Crash

Another method of detecting (and preventing) spear phishing attacks is to deploy the Enhanced Mitigation Experience Toolkit (EMET). EMET is a free Microsoft developed utility that is designed to aid in the prevention of many common memory manipulation based attacks such as the previously described buffer overflow attack. When EMET detects an attack it crashes the application to prevent the exploit from being successful. EMET will log this as an error message (EventID 2) and may, if configured to do so, display a ***pop-up*** notification to the end user. EMET however, does not have a centralized management console and a third-party log management solution should be used to collect these events.

Another possible indicator of an intrusion as referenced by the SANS Intrusion Discovery Cheat Sheet (SANS, n.d.) is a Windows File Protection (WFP) warning event (EventID 64004). The windows file protection service monitors critical system files

such as executables (.exe) and dynamic link libraries (.dll) and attempts to prevent unauthorized software from modifying or replacing these files (Microsoft, n.d.). The windows file protection service will generate a warning if it detects a change to a critical system file and is unable to restore it to a known good state.

One often overlooked way of detecting an intrusion is by paying attention to anti-virus alerts. In many enterprises, anti-virus alerts are rarely monitored. Context can be extremely valuable here. Was the alert generated when an email attachment was opened possibly indicating a spear phish or was it a more mundane drive-by web surfing type attack? While it may not be feasible to investigate every anti-virus alert, it may be possible to correlate certain new process events with anti-virus alerts to add some context and value. For example, Internet Explorer being launched followed directly by an anti-virus alert suggests that the user clicked on a malicious hyperlink from an email or that the users home page is serving up malicious code. Another example would be an anti-virus alert followed by a Least Frequency of Occurrence (LFO) ***new process created*** event. This may indicate that an attack against the host was successful and that the anti-virus only detected one in a series of attacks against the client.

3.2 Detecting Persistence.

One of the goals of every attacker is to establish a foothold to maintain control of a compromised computer even if the user logs off or if the computer is rebooted (Skoudis & Zeltser 2004). This control is referred to as ***persistence***. There are a many ways to create a persistent service on a computer. Some of the more common ways are as follows:

- Create and install a new service. (EventID 601|4697)
- Create a new scheduled task. (EventID 602|4698)
- Modify the registry keys so the service is started at boot. (EventID 567|4657)

Detecting changes made to registry keys requires two things. First, Auditing for File and Object Access must be globally enabled. Secondly, auditing must be enabled on the individual registry keys of interest. The following registry keys listed on the SANS

Intrusion Discovery Cheat Sheet (SANS, n.d.) for Windows are commonly used by malware (and legitimate programs) to create persistence.

- HKLM\Software\Microsoft\Windows\CurrentVersion\Run
- HKLM\Software\Microsoft\Windows\CurrentVersion\Runonce
- HKLM\Software\Microsoft\Windows\CurrentVersion\RunonceEx
- HKCU\Software\Microsoft\Windows\CurrentVersion\Run
- HKCU\Software\Microsoft\Windows\CurrentVersion\Runonce
- HKCU\Software\Microsoft\Windows\CurrentVersion\RunonceEx

"One of the ways in which malicious code attempts to protect its turf is by disabling the virus protection mechanisms on the target machine" (Skoudis & Zeltser, 2004). Here are some examples of what you might look for:

- The <protection mechanism service > terminated unexpectedly.* (EventID 7034)
- The <protection mechanism service > was successfully sent a .* (EventID 7035)
- The <protection mechanism service > entered the stopped state.* (EventID 7036)
- The <protection mechanism service >service was change from.* (EventID 7040)

Some examples of the <protection mechanism service > would include an enterprise anti-virus solution, an enterprise host based intrusion prevention system (HIPS), Windows Defender or an enterprise Windows syslog agent. While it may be common for some of these services to restart during upgrades or even during signature updates, it would be very unlikely and suspicious if anti-virus and Windows Defender were both stopped.

3.3 Detecting Privilege Escalation

Escalating privileges involves acquiring access to resources across the network. One very common privilege escalation technique is called "***Pash the Hash***". Pash the Hash is a two step process that involves dumping the locally cached credentials called the hash and using it to remotely login to other systems across the network (the pass). Dumping the hash is a trivial process that requires three things, A hash dumping tool kit, local administrative privileges and the debug user right. The debug user right assignment

is given to the built-in local administrator account by default. It is a privilege that is seldom used by legitimate software. One strategy to detect the dumping of the hash would be to remove the debug privilege from your environment, allow it by exception as needed, and then monitor for any changes (EventID 608|4704) to the privilege (Hummel, 2009).

3.4 Detecting Internal Reconnaissance

Once a system has been compromised, attackers often use built in Windows command line tools such as ***ipconfig***, ***tasklist***, and ***net use*** to learn as much as possible about the environment. While some system administrators and power users use these tools on a regular basis, most end users are unlikely to use these tools. Detecting internal reconnaissance may be possible by profiling ***who*** normally uses these tools, ***what*** tools are used and ***how*** the tools are used. For example, if a network engineer is troubleshooting network issues, he/she may issue repeated ***ipconfig***, ***nslookup*** and ***ping*** commands. This would be a very different pattern of usage than a single ***ipconfi***g command followed by a single ***tasklist*** command followed by multiple ***net use*** commands. As Mandiant (Mcwhorter, 2013) has pointed out, some attackers use batch scripts to automate internal reconnaissance. Scripts can generate patterns that should easily stand out from normal patterns.

3.5 Detecting Lateral Movement

Once an attacker has compromised a system, established persistence, and gained legitimate credentials by dumping the hash, he\she can then move laterally across the network to other resources to gather information or to install software on other systems. This is the second part of the pass the hash attack described earlier. Because the attacker is using valid credentials to authenticate over the network, this type of activity can be difficult to detect. One method of detection would be to search for unusual host to host network based logins. Windows logs these events at Type 3 Network logins or if using Remote Desktop Protocol RDP then Type 10 Network logins (EventID 528|529|4624|4625).

Another method to detect lateral movement is the detection of unusual accounts attempting network based logins. For example, many companies use the default built-in administrator account as a last resort account for field admins that will provide local login access when all else fails. As this type of account is for authenticating locally, any logins over the network using this account should be considered very suspicious.

4. Conclusion

Collecting Windows event logs can be an extremely valuable resource as part of a larger security monitoring process. Workstations, and the end users behind them, have become THE target in today's ongoing cyber war. Workstations are where the initial compromises happen. Workstations are where the APT builds bridge heads to attack companies from within to stealthily collect and steal our data. In short, workstations are where the action is at. It is therefore critical to collect and analyze logs produced by these systems. In order to collect and consume this huge volume of logs and feed them to already overwhelmed SIEM infrastructures, a new strategy is required. This strategy is called an output driven strategy where only those events that will later be used to create an alert or a report will be collected. Workstation event logs can be analyzed using some of the same techniques currently used to look for malware during memory analysis. Event logs can also be used to create an invaluable historical record that can later be parsed as new IOCs are created and shared. One of the most promising ways to use workstation event logs could potentially be to create IOCs based on patterns of events which together form a behavior profile. Attackers like all people have habits, routines and techniques. They prefer some commands to others, and tend to run those commands in certain orders and in certain ways. If these ***Tactics, Techniques*** and ***Procedures*** (***TTPs***) can be discerned, they may produce high quality IOCs that can be used over long periods of time which will ultimately be very costly to the attackers (Bianco, 2013).

Russ Anthony, ranthony00@gmail.com

5. References

Bianco, D. (2013). *The Pyramid of Pain*. Retrieved from Enterprise Detection & Response:http://detect-respond.blogspot.com/?view=classic#!/2013/03/the-pyramid-of-pain.html

Chuvakin, A. (2012). *On "Output-driven" SIEM*. Retrieved from Gartner: http://blogs.gartner.com/anton-chuvakin/2012/09/24/on-output-driven-siem/

Carvey, H. (2010). *Thoughts on APT*. Retrieved from Windows Incident Response: http://windowsir.blogspot.com/2010/01/thoughts-on-apt.html

Hummel, C. (2009). *Why Crack When You Can Pass the Hash?* Retrieved from http://www.sans.org/reading_room/whitepapers/testing/crack-pass-hash_33219

Kornblum, J. (2010). *Windows Memory Analysis*. Retrieved from http://jessekornblum.com/presentations/usna10.pdf

Mcwhorter, D. (2013). *APT1: Exposing One of China's Cyber Espionage Units*. Retrieved from Mandiant: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

Gerhards, R. (2009). *The Syslog Protocol*. (Request for Comments: 5424) Retrieved from IETF: http://tools.ietf.org/html/rfc5424

SANS (n.d.). *SANS Institute Intrusion Discovery Cheat Sheet v2.0*. Retrieved from SANS Institute: http://www.sans.org/score/checklists/ID_Windows.pdf

Skoudis, E., & Zeltser, L. (2004). *Malware: Fighting malicious code*. Upper Saddle River, NJ: Prentice Hall PTR.

Torres, A. (2013). *Windows Memory Forensics In-Depth*. Sans FOR 526. Lecture conducted from Orlando, FL.

I owe general credit to the SANS courses and presentations for information processes and ideas that have over time become part of my standard way of thinking, if not specifically quoted in the paper.

SANS Windows Memory Forensics In-Depth (2013)
SANS Network Forensics (2012)
SANS Incident Handling (2011)
SANS Security Essentials (2010)
SANS Network Penetration Testing and Ethical Hacking (2010)
SANS Reverse Engineering Malware (2009)
SANS + S Training Program for the CISSP Certification EXAM (2008)
SANS Intrusion Detection In-depth (2007)
SANS Computer Forensics, Investigation, and Response (2006)
SANS Hacker Techniques, Exploits and Incident Handling (2003)
SANS Firewalls, Perimeter Protections and VPN's (2003)

Russ Anthony, ranthony00@gmail.com

Appendix 1: Windows XP regular expressions

Windows XP regular expressions	Source	EventID Number
".*The exception generated was.*"	Application	4097
".*he protected system file.*"	Application	64004
".*EMET_DLL Module logged the following event:.*"	Application	2
".*your anti-virus alerts.*"	Application	varies
".*A new process has been created:.*"	Security	592
".*Attempt to install service.*"	Security	601
".*Scheduled Task created.*"	Security	602
".*Logon Type: (3|10).*"	Security	528, 529
".*\\Software\\Microsoft\\Windows\\CurrentVersion\\Run.*"	Security	567
".*service terminated unexpectedly\..*"	System	7034
".*service was successfully sent a.*"	System	7035
". *service entered the.*"	System	7036
".*service was changed from.*"	System	7040

Table3: Windows XP regular expression white list

Appendix 2: Open Source Solution

Appendix 2 details a simple, low cost solution to gather Windows Events using open source tools. These tools include ***eventlog-to-syslog*** as the windows syslog agent, ***rsyslog*** as the centralized collector, and ***Simple Event Correlator (SEC)*** for the analysis and alerting. Diagram 3 shows the overall architecture with flow. The overall process consists of the following steps.

Step 1: Make sure that event logging has been enabled on the Windows workstations in the environment via the audit policy. Once this has been enabled, event logs can be seen in the Windows event viewer.

Step2: Install the windows syslog agent called **"*eventlog-to-syslog"*** on the Windows workstations. This tool takes the windows events, converts them to the syslog format and sends them across the network to a centralized syslog collector. **E*ventlog-to-syslog*** can also filter Windows events based on the Event ID. Appendix 2.1 contains a configuration file with Event ID filters to get started with.

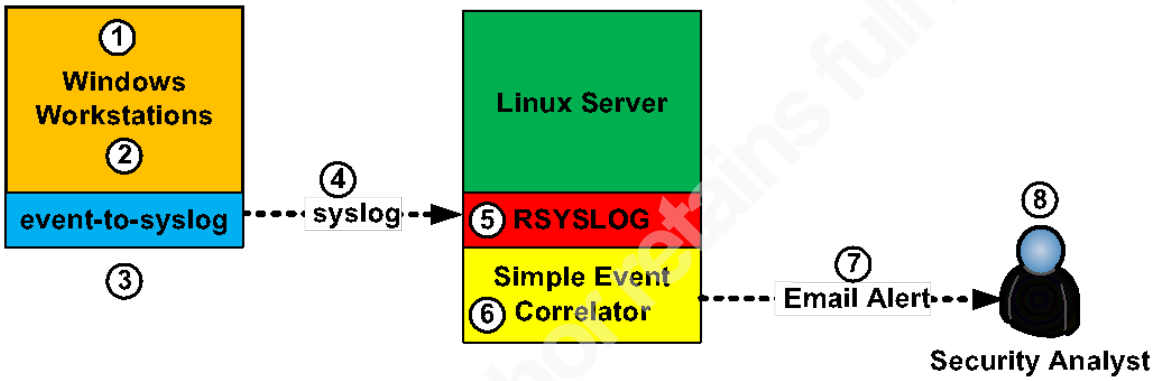
Step 3: Build a centralized syslog collector using a Linux server and install the tool called ***rsyslog*****. Rsylog** will collect the Windows events off the wire and write them to a log file. The log file can be compressed and archived using the built-in Linux tool called **logrotate**. Appendix 2.2 contains an **rsyslog** configuration file that can be used to stand up a centralized syslog collector.

Step 4: Install the tool called ***Simple Event Correlator (SEC)*** on the same Linux server that **rsyslog** was installed on. **SEC** will monitor the log files created by ***rsyslog*** and will create an email alert when an interesting event has been detected. Appendix 2.3 contains a **SEC** configuration file that can be used to detect interesting events.

Step 5: Test to make sure steps 1 through 4 have been done properly. Create and execute test scenarios that create interesting events to ensure that the correct events logs are created, collected, written to a file, and that email alerts are generated.

Russ Anthony, ranthony00@gmail.com

Step 6: Monitor and tune the **SEC** rule set as needed for the environment. Use "suppress" rules to tune out noise. Build and deploy new alert rules based on new threats and vulnerabilities as needed.



1. A new malicious process starts (for example: scvhost.exe Event ID 4688).
2. Windows auditing creates an Event ID 4688 and writes it to the local Windows event log.
3. The Event ID matches the white list filter in Event-to-syslog.
4. The matched log is converted to syslog and sent across the wire to the centralized rsyslog server.
5. Rsyslog collects the log over the wire and writes it to /var/log/win7.log.
6. Simple Event Correlator monitors the /var/log/win7.log file and makes a match on the string "scvhost" (SEC rule number 01-00).
7. An email alert is created and sent to the security analyst.
8. Security analyst receives email alert.

Diagram 3 – Open Source Logging Architecture

Appendix 2.1 Eventlog-to-syslog

Install Notes

- http://code.google.com/p/eventlog-to-syslog/
- must be installed in C:\Windows\System32
- to install run: evtsys -i -n -h logcollector.company.com
- -i Install service

Russ Anthony, ranthony00@gmail.com

- -h host Name of log host
- -n Include only those events specified in the config file. (White list)
- net start evtsys (start the service which will fail but will create database)
- edit evtsys.cfg
- net start evtsys (start again

Configuration file (C:\windows\system32\evtsys.cfg)

Add these to the end of the cfg file to create your white list (the first cut)
Format is Source:EventID

WinXP Configuration File

Security:528
Security:529
Security:567
Security:592
Security:601
Security:602
Security:608
Security:612
Security:636
Service Control Manager:7034
Service Control Manager:7035
Service Control Manager:7036
Service Control Manager:7040
DrWatson:4097
Windows File Protection:64004
EMET:2
WinDefend:3005
Your Anti-Virus:

Win7 Configuration File

Security:4624
Security:4625
Security:4657
Security:4688
Security:4697
Security:4698
Security:4704
Security:4719
Security:4732
Service Control Manager:7034
Service Control Manager:7035
Service Control Manager:7036

Russ Anthony, ranthony00@gmail.com

Service Control Manager:7040
Windows Error Reporting:1001
Windows File Protection:64004
EMET:2
Windows Defender:1006
Your Anti-Virus:

This completes setting up the syslog agent so that events will be forwarded to your collector.

Appendix 2.2 Rsyslog

Install Notes

- Rsyslog is the centralized event collector.
- http://www.rsyslog.com/
- Rsyslog support via Adiscon is Excellent!
- This configuration file will write all Windows Events to either the win7.log or the winxp.log in the /var/log directory.

Configuration file (/etc/rsyslog.conf)

```
# rsyslog configuration file
# For more information see /usr/share/doc/rsyslog-*/rsyslog_conf.html
# If you experience problems, see http://www.rsyslog.com/doc/troubleshoot.html
#Provides UDP syslog reception
 module(load="imudp" MaxSessions="10000")
 #collect win7 events on UDP port 514
 input(type="imudp" port="514" Ruleset="win7")
 #collect winxp events on UDP port 515
 input(type="imudp" port="515" Ruleset="winxp")

#### GLOBAL DIRECTIVES ####
# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf
```

Russ Anthony, ranthony00@gmail.com

```
#### RULES ####
ruleset(name="win7"){
        action(type="omfile" file="/var/log/win7.log")
                        }
ruleset(name="winxp"){
        action(type="omfile" file="/var//log/winxp.log")
                        }
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                 /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none        /var/log/messages

# The authpriv file has restricted access.
authpriv.*                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                  /var/log/maillog

# Log cron stuff
cron.*                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                 :omusrmsg:*

# Save news errors of level crit and higher in a special file.
uucp,news.crit                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                /var/log/boot.log
```

Russ Anthony, ranthony00@gmail.com

Appendix **2.3 Simple Event Correlator (SEC):**

Install Notes

- http://simple-evcorr.sourceforge.net/
- http://simple-evcorr.sourceforge.net/SEC-tutorial/article.html
- The commands to launch SEC are as follows:
- /usr/bin/perl -w ./sec -conf=win7sec.conf -input=/var/logs/win7.log
- /usr/bin/perl -w ./sec -conf=winxpsec.conf -input=/var/logs/winxp.log
- This configuration will monitor the log files created by Rsyslog in the /var/log directory (winxp.log and win7.log)

Configuration file for winxp (winxp.conf)

```
# Rule Number 00-00
# Suppress rules suppress the matched pattern
# Adding "(?i)" makes the regex case insensitive
# Suppress noisy events that are of no interest to us (the second cut)
# Event ID 592 or other
type=Suppress
ptype=RegExp
pattern=(?i)C:\\Windows\\System32\\search(protocolhost|filterhost|indexer).exe
desc=$0


# Begin to look for suspicious processes
# Rule Number 01-00
# Single rules perform an action if the pattern is matched
# Alert on purposely misspelled process
# Event ID 592
type=Single
ptype=RegExp
pattern=(?i)\\(scvhost|svcdost|scvdost|iexplorer)\.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Misspelled Process" admin@company.com

# Rule Number 01-01
```

Russ Anthony, ranthony00@gmail.com

```
# Alert on lower case drive letters
# Event ID 592
type=Single
ptype=RegExp
pattern=(File Name: (c|d|e):\\)
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Misspelled Process" admin@company.com

# Rule Number 01-02-A
# Suppress svchost.exe if the path is correct.
# Event ID 592
type=Suppress
ptype=RegExp
pattern=(?i)(C:\\WINDOWS\\System32\\svchost.exe)
desc=$0

# Rule Number 01-02-B
# Alert on svchost.exe with an incorrect patch
# Event ID 592
type=Single
ptype=RegExp
pattern=(?i)\\svchost.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Incorrect Path" admin@company.com

# Rule Number 01-03-A
# Suppress explorer.exe if the path is correct.
# Event ID 592
type=Suppress
ptype=RegExp
pattern=(?i)C:\\WINDOWS\\explorer.exe
desc=$0

# Rule Number 01-03-B
# Alert on explorer.exe with an incorrect patch
# Event ID 592
type=Single
ptype=RegExp
pattern=(?i)\\explorer.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Incorrect Path" admin@company.com

# Rule Number 01-04
# Alert on a crashed application
# EventID 4097
type=Single
```

Russ Anthony, ranthony00@gmail.com

```
ptype=RegExp
pattern=(?i)(Adobe|Microsoft Office|Java|wmplayer).*(EventID 4097)
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Application Crash" admin@company.com

# Rule Number 01-05
# Alert on EMET event
# EventID 2
type=Single
ptype=RegExp
pattern=EMET_DLL Module logged the following event
desc=$0
action=pipe '$0' /bin/mailx -s "EMET Alert" admin@company.com

# Rule Number 01-06
# Alert on Windows File Protection
# EventID 64004
type=Single
ptype=RegExp
pattern=(?i)EventID 64004
desc=$0
action=pipe '$0' /bin/mailx -s "Windows File Protection Alert" admin@company.com

# Rule Number 01-07
# Alert on Windows Defender or other Anti-Virus
# EventID 3005
type=Single
ptype=RegExp
pattern=Windows Defender has detected
desc=$0
action=pipe '$0' /bin/mailx -s "Windows Defender Alert" admin@company.com

# Rule Number 01-08-A
# Suppress allowed screen saver  files ( or .com files )
# Event ID 592
type=Suppress
ptype=RegExp
pattern=C:\\WINDOWS\\system32\\(scrnsave.scr|ss3dfo.scr|ssbezier.scr|ssflwbox.scr|ssm
arque.scr|ssmypics.scr|ssmyst.scr|sspipes.scr|ssstars.scr|sstext3d.scr)
desc=$0


#rule Number 01-08-B
# Alert on non standard Screen saver file
# Event ID 592
type=Single
```

Russ Anthony, ranthony00@gmail.com

```
ptype=RegExp
pattern=(?i)File Name:.*\.scr
desc=$0
action=pipe '$0' /bin/mailx -s "Malicious File Extension Detected" admin@company.com


# Rule Number 01-09
# Alert on suspicious toolsets
# Event ID 592
type=Single
ptype=RegExp
pattern=(?i)(\\win32dd.exe|\\win64dd.exe|\\Cachedump|\\Fgdump|\\gsecdump|\\Lslsass|\\
mimikatz|\\PwDump7|\\pwdumpX|\\pwdump|\\wce.exe|\\getlsasrvaddr)
desc=$0
action=pipe '$0' /bin/mailx -s "Suspicious Tool Alert" admin@company.com

# Rule Number 01-10
# Alert on Anti-virus Event
# EventID varies
type=Single
ptype=RegExp
pattern=(virus found)
desc=$0
action=pipe '$0' /bin/mailx -s "Anti-VirusAlert" admin@company.com

# Begin to detect persistence
# Rule Number 02-00
# Alert on new service
# EventID 601
type=Single
ptype=RegExp
pattern=EventID 601
desc=$0
action=pipe '$0' /bin/mailx -s "New Service Installed" admin@company.com

# Rule Number 02-01
# Alert on new scheduled task
# EventID 602
type=Single
ptype=RegExp
pattern=EventID 602
desc=$0
action=pipe '$0' /bin/mailx -s "New Scheduled Task" admin@company.com

# Rule Number 02-02
# Alert on persistence keys
```

```
# Event ID 567
type=Single
ptype=RegExp
pattern=(?i)\\Software\\Microsoft\\Windows\\CurrentVersion\\Run
desc=$0
action=pipe '$0' /bin/mailx -s "Persistence Key Alert" admin@company.com

# Rule Number 02-03
# Alert on service change
# EventID (7034|7035|7036|7040)
type=Single
ptype=RegExp
pattern=(your-anti-virus|EMET|Defender).*EventID (7034|7035|7036|7040)
desc=$0
action=pipe '$0' /bin/mailx -s "Protection Disabled" admin@company.com

# Rule Number 02-04-A
# Suppress Policy Changes made by SYSTEM
# Event ID 612
type=Suppress
ptype=RegExp
pattern=(?i)NT AUTHORITY\\SYSTEM:.*Audit Policy Change:
desc=$0

# Rule Number 02-04-B
# Alert on Audit Policy Change
# EventID 612
type=Single
ptype=RegExp
pattern=EventID (612)
desc=$0
action=pipe '$0' /bin/mailx -s "Audit Policy Change" admin@company.com



# Begin to detect Privilege Escalation
# Rule Number 03-00
# Alert on User Right Assigned
# EventID 608
type=Single
ptype=RegExp
pattern=EventID (608)
desc=$0
action=pipe '$0' /bin/mailx -s "User Right Assigned Alert" admin@company.com

# Rule Number 03-01
```

```
# Alert on Accounts being added to the Local Administrators Group
# Event ID 636
type=Single
ptype=RegExp
pattern=(Security Enabled Local Group Member Added.*BUILTIN\\Administrators)
desc=$0
action=pipe '$0' /bin/mailx -s "Local Account Added to Administrators"
admin@company.com


# Begin to detect Lateral Movement
# Rule Number 04-00
# Alert on Type 3 and Type 10 Network logins
# Likely to be noisy need to baseline
# EventID (528|529)
type=Single
ptype=RegExp
pattern=EventID (528|529)
desc=$0
action=pipe '$0' /bin/mailx -s "Network Login Attempt" admin@company.com

# Rule Number 04-01
# Alert on Local Admin Account attempting Network Login.
# EventID (528|529)
type=Single
ptype=RegExp
pattern=(?i)User Name: (<local administrator name>).*Logon Type: (3|10)
desc=$0
action=pipe '$0' /bin/mailx -s "Lateral Attempt Alert" admin@company.com
```

Configuration file for win7 (win7.conf)

```
# Win7 Rule set
# Rule Number 00-00
# Suppress rules suppress the matched pattern
# Adding "(?i)" makes the regex case insensitive
# Suppress noisy events that are of no interest to us (the second cut)
# Event ID 4688 or other
type=Suppress
ptype=RegExp
pattern=(?i)C:\\Windows\\System32\\Search(ProtocolHost|FilterHost|Indexer).exe
desc=$0
```

Russ Anthony, ranthony00@gmail.com

```
# Begin to look for suspicious processes
# Rule Number 01-00
# Single rules perform an action if the pattern is matched
# Alert on purposely misspelled process
# Event ID 4688
type=Single
ptype=RegExp
pattern=(?i)\\(scvhost|svcdost|scvdost|iexplorer)\.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Misspelled Process" admin@company.com

# Rule Number 01-01
# Alert on lower case drive letters
# Event ID 4688
type=Single
ptype=RegExp
pattern=(File Name: (c|d|e):\\)
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Misspelled Process" admin@company.com

# Rule Number 01-02-A
# Suppress svchost.exe if the path is correct.
# Event ID 4688
type=Suppress
ptype=RegExp
pattern=(?i)(C:\\WINDOWS\\System32\\svchost.exe)
desc=$0

# Rule Number 01-02-B
# Alert on svchost.exe with an incorrect patch
# Event ID 4688
type=Single
ptype=RegExp
pattern=(?i)\\svchost.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Incorrect Path" admin@company.com


# Rule Number 01-03-A
# Suppress explorer.exe if the path is correct.
# Event ID 4688
type=Suppress
ptype=RegExp
pattern=(?i)(C:\\WINDOWS\\explorer.exe)
desc=$0
```

```
# Rule Number 01-03-B
# Alert on explorer.exe with an incorrect patch
# Event ID 4688
type=Single
ptype=RegExp
pattern=(?i)\\explorer.exe
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Incorrect Path" admin@company.com

# Rule Number 01-04
# Alert on a crashed application
# Event ID 1001
type=Single
ptype=RegExp
pattern=(?i)(Adobe|Microsoft Office|Java|wmplayer).*(EventID 1001)
desc=$0
action=pipe '$0' /bin/mailx -s "Alert Application Crash" admin@company.com

# Rule Number 01-05
# Alert on EMET event
# Event ID 2
type=Single
ptype=RegExp
pattern=EMET_DLL Module logged the following event
desc=$0
action=pipe '$0' /bin/mailx -s "EMET Alert" admin@company.com

# Rule Number 01-06
# Alert on Windows File Protection
# Event ID 64004
type=Single
ptype=RegExp
pattern=(?i)EventID 64004
desc=$0
action=pipe '$0' /bin/mailx -s "Windows File Protection Alert" admin@company.com

# Rule Number 01-07
# Alert on Windows Defender or other Anti-Virus
# Event ID 1006
type=Single
ptype=RegExp
pattern=Windows Defender has detected
desc=$0
action=pipe '$0' /bin/mailx -s "Windows Defender Alert" admin@company.com
```

```
# Rule Number 01-08-A
# Suppress allowed screen saver  files ( or .com files )
# Event ID 4688
type=Suppress
ptype=RegExp
pattern=C:\\WINDOWS\\system32\\(scrnsave.scr|Bubbles.scr|Bubbles.scr|PhotoScreensa
ver.scr|Ribbons.scr|ssText3d.scr)
desc=$0

# Rule Number 01-08-B
# Alert on non standard Screen saver file
# Event ID 4688
type=Single
ptype=RegExp
pattern=(?i)File Name:.*\.scr
desc=$0
action=pipe '$0' /bin/mailx -s "Malicious File Extension Detected" admin@company.com

# Rule Number 01-09
# Alert on suspicious toolsets
# Event ID 4688
type=Single
ptype=RegExp
pattern=(?i)(\\win32dd.exe|\\win64dd.exe|\\Cachedump|\\Fgdump|\\gsecdump|\\Lslsass|\\
mimikatz|\\PwDump7|\\pwdumpX|\\pwdump|\\wce.exe|\\getlsasrvaddr)
desc=$0
action=pipe '$0' /bin/mailx -s "Suspicious Tool Alert" admin@company.com

# Rule Number 01-10
# Alert on Anti-virus Event
# EventID varies
type=Single
ptype=RegExp
pattern=(you-anti-virus)
desc=$0
action=pipe '$0' /bin/mailx -s "Anti-Virus Alert" admin@company.com

# Rule Number 02-00
# Alert on new service
# Event ID 4697
type=Single
ptype=RegExp
pattern=EventID 4697
desc=$0
action=pipe '$0' /bin/mailx -s "New Service Installed" admin@company.com
```

```
# Rule Number 02-01
# Alert on new scheduled task
# Event ID 4698
type=Single
ptype=RegExp
pattern=EventID 4698
desc=$0
action=pipe '$0' /bin/mailx -s "New Scheduled Task" admin@company.com

# Rule Number 02-02
# Alert on persistence keys
# Event ID 4657

type=Single
ptype=RegExp
pattern=(?i)\\Software\\Microsoft\\Windows\\CurrentVersion\\Run
desc=$0
action=pipe '$0' /bin/mailx -s "Persistence Key Alert" admin@company.com

# Rule Number 02-03
# Alert on service change
# EventID (7034|7035|7036|7040)
type=Single
ptype=RegExp
pattern=(your-anti-virus|EMET|Defender).*EventID (7034|7035|7036|7040)
desc=$0
action=pipe '$0' /bin/mailx -s "Protection Disabled" admin@company.com

# Rule Number 02-04-A
# Suppress Policy Changes made by SYSTEM
# EventID 4719
type=Suppress
ptype=RegExp
pattern=(?i)NT AUTHORITY\\SYSTEM:.*Audit Policy Change:
desc=$0

# Rule Number 02-04-B
# Alert on Audit Policy Change
# EventID 4719
type=Single
ptype=RegExp
pattern=EventID (4719)
desc=$0
action=pipe '$0' /bin/mailx -s "Audit Policy Change" admin@company.com

# Begin to detect Privilege Escalation
```

```
# Rule Number 03-00
# Alert on User Right Assigned
# EventID 4704
type=Single
ptype=RegExp
pattern=EventID (4704)
desc=$0
action=pipe '$0' /bin/mailx -s "User Right Assigned Alert" admin@company.com

# Rule Number 03-01
# Alert on Accounts being  added to the Local Administators Group
# EventID 4732

type=Single
ptype=RegExp
pattern=(Security Enabled Local Group Member Added.*BUILTIN\\Administrators)
desc=$0
action=pipe '$0' /bin/mailx -s "Local Account Added to Administrators"
admin@company.com

# Begin to detect Lateral Movement
# Rule Number 04-00
# Alert on Type 3 and Type 10 Network logins
# going to be Noisy, need to baseline
# EventID (4624|4625)
type=Single
ptype=RegExp
pattern=EventID (4624|4625)
desc=$0
action=pipe '$0' /bin/mailx -s "Network Login Attempt" admin@company.com

# Rule Number 04-01
# Alert on Local Account attempting Network Login.
# EventID (4624|4625)
type=Single
ptype=RegExp
pattern=(?i)User Name: (<local administrator name>).*Logon Type: (3|10)
desc=$0
action=pipe '$0' /bin/mailx -s "Lateral Attempt Alert" admin@company.com
```