Log Correlation Engine Log Normalization Guide

December 22, 2014

(Revision 2)

Table of Contents

Introduction

This document discusses log analysis for Tenable Network Security's **Log Correlation Engine**. Please email any comments and suggestions to support@tenable.com.

It is assumed that the reader has the following prerequisite skills:

- A working knowledge of Secure Shell (SSH) and key exchange.
- Experience with the Unix operating system, including regular expressions.
- Familiarity with Tenable's Log Correlation Engine (LCE) operation and architecture as described in the "Log Correlation Engine Administration and User Guide".
- Knowledge of general log formats from various operating systems, network devices and applications.

Standards and Conventions

Throughout the documentation, filenames, daemons, and executables are indicated with a **courier bold** font such as **gunzip**, **httpd**, and **/etc/passwd**.

Command line options and keywords are also indicated with the **courier bold** font. Command line examples may or may not include the command line prompt and output text from the results of the command. Command line examples will display the command being run in **courier bold** to indicate what the user typed while the sample output generated by the system will be indicated in courier (not bold). Following is an example running of the Unix **pwd** command:

```
# pwd
/opt/lce/
#
```

Important notes and considerations are highlighted with this symbol and grey text boxes.

Tips, examples, and best practices are highlighted with this symbol and white on blue text.

Log Parsing and Normalization

Architecture

The LCE daemon, **lced**, receives entire log messages from **syslog** messages or LCE clients. For example, the following log message could arrive at the LCE through one of the LCE's log parsing clients or have been sent via **syslog** across the network directly:

```
Jun 22 01:02:34 godzilla kernel: eth1: Setting promiscuous mode.
```

Normalization

When the LCE receives these logs, it uses libraries (known as **.prm** files) to analyze the message. The first part of analysis is to recognize the message type with unique keywords. Once a message type is recognized, it is normalized and relevant data is populated into an event schema.

Normalization types do not change based on the LCE product releases; they are based entirely on the content update cycle. Therefore, this list is subject to change.

Type	Description
access-denied	Flags attempts to retrieve objects, files, network shares and other resources that are denied. These events are distinct from authentication failures, blocked firewall connections and attempts to access web pages that do not exist that are respectively normalized to the **`login-failure`**, **`firewall`** and **`web-error`** event types. Examples: • **Juniper-Command_Execution_Error** – A user was prevented from running a command. • **FTP-Directory_Create_Attempt** – A user was denied in an attempt to create a directory on an FTP server. • **Bind-Denied_Version_Query** – An attempt to perform a version query against a BIND DNS server was denied. • **Windows-Privileged_Object_Operation_Failure** – An attempt to access a Windows Object was denied.
application	Denotes logs from any application such as Nessus, Symantec AntiVirus, SecurityCenter, the WU-FTP server, Exchange, sendmail, etc. that is noteworthy but not indicative of an error, a login failure, a connection, a restart of the application, an operating system event or a major function of the device. For example, sendmail logs that indicate authenticated users are normalized to the **`login`** event type and sendmail logs that indicate spam email are logged to the **`spam`** event type. Examples: • **BIND-Transfer** – A DNS server delivered a zone transfer. • **Filezilla-Directory_Listing** – A Windows-based FTP server had a user perform a directory listing. • **Sendmail-Verify_User_Attempt** – The sendmail application encountered an attempt to verify the existence of a user account. • **MYSQL-Total_Allocated_Space** – A MySQL database logged its allocated amount of disk space. • **MSSQLSVR-Database_Unfrozen** – An MS SQL server became unfrozen. • **Nessus-Scan_Started** – A Nessus scan started. • **Windows-Defender_Scan_Started** – Windows Defender started a virus scan.
connection	Notes any type of audited network connection that is not directly logged via the Tenable NetFlow Monitor (TFM) or the Tenable Network Monitor (TNM). Event sources include allowed connections through firewalls, established VPN sessions and connections by some types of applications. TFM and TNM events are normalized to the **`network`** event type. Examples: • **Checkpoint-Accepted_UDP** – A Check Point firewall has allowed and logged a UDP network session. • **IRC-Chat_Connect** – An IRC daemon has logged a connection to the chat server. • **CiscoASA-Built_Outbound_TCP_Connection** – A Cisco ASA firewall has allowed and logged an outbound TCP connection. • **FTP-Connection** – An FTP server has logged a connection. • **Sendmail-Bad_Connection_Termination** – The sendmail server had a connection that needed to be terminated.

continuous	The LCE can identify hosts that are generating certain specific event types for periods of 20 minutes or longer. For example, a host may be infected with a worm and attack small numbers of targets every five minutes. Examples: • **Long_Term_DNS_Failures** – A host is likely performing vulnerability scans, attempting to send large volumes of spam email, or has its DNS information misconfigured as it has been encountering DNS lookup errors for periods of 20 to 120 minutes continuously. • **Long_Term_Error_Activity** – A host has had errors reported from it for periods of 20 to 120 minutes continuously, which can indicate a major problem. • **Long_Term_Network_Scanning** – A host has had port scan events reported from it for periods of 20 to 120 minutes continuously.
database	Denotes logs generated by PVS from observed SQL queries. As PVS monitors PostgreSQL, Oracle, MySQL, MS SQL, etc. network transactions, it creates logs that indicate a variety of database actions such as inserts and select statements. Examples: • **PVS-Database_INSERT_Command** – The PVS has detected an INSERT event into the database. • **PVS-Database_CREATE_Command** – The PVS has detected a CREATE event into the database.
data-leak	Flags logs from the PVS or other Data Leak Prevention products that indicate the presence of sensitive data such as a credit card or Social Security number. PVS must be specifically configured with DLP rules available from the Tenable Support Portal. Examples: • **PVS-Credit_Card_Detection** – The PVS has detected network content that contained a credit card number. • **iGuard-Skintone_Image** – The McAfee DLP product detected an image that likely contains human skin tones and could be related to adult content.
detected-change	The LCE automatically recognizes many types of `system` events that indicate change and creates secondary higher level events. These events can be used to ease reporting, alerting and creating dashboards. Examples: • **New_MAC** – A new Ethernet address was encountered that had not been previously seen. • **PVS-New_Port_Browsing** – The PVS has detected a host that is browsing the network or Internet on a new port. • **Router_Change** – A configuration change to a router was encountered. • **Software_Installed** – Some sort of application or package was installed.

dhcp	Logs from DHCP servers that indicate new leases are given the **`dhcp`** event type. Any type of errors, login profiles, system status messages, etc. are normalized to other LCE event types. These events solely focus of monitoring DHCP activity. Examples: • **DLink-Network_Computer_Assigned_IP** – A home D Link router issued a DHCP lease. • **DHCP-Request** – A generic DHCP lease request was received. • **Fortinet-DHCP_Config_Offer** – A Fortinet firewall offered a DHCP lease to a host.
dns	Denotes any type of log from a DNS server or from real-time network monitoring by the PVS that indicates a DNS query or a DNS query lookup failure. LCE summary information as well as Fast Flux detection is also logged here. Examples: • **Bind-Query_IPv6** – The Bind DNS server logged an IPv6 address request. • **Domain_Summary** – The LCE has summarized unique domain names queried for a host. • **PVS-DNS_Client_Query** – The PVS sniffed a DNS query and logged it. • **Fluxing_Domain_Detected** – The LCE has detected a domain in use that is likely part of a “Fast Flux” botnet.
dos	Denotes any type of attempt to perform a denial of service attempt logged by a network IDS, an application, a firewall, an NBAD or even an operating system. Examples: • **CiscoPIX-Deny_IP_Teardrop_Fragment** – A Cisco PIX firewall recognized a teardrop denial of service event. • **Foundry-Fragmentation_DOS** – A Foundry switch encountered packet fragmentation that is likely a denial of service attack. • **Snort-TCP_DOS_Attack** – The Snort IDS has monitored a potential denial of service attack. • **NetscreenIDP-DDOS_Activity** – Communication with a distributed denial of service (DDoS) attack has been observed from a Juniper Netscreen IDP device.
error	Denotes any type of system, application, router or switch log that indicates some sort of error. Logs that indicate crashes and hung process are sent to the **`process`** event type. Examples: • **CiscoWireless-Config_Error** – A Cisco Wireless AP has encountered a configuration error. • **CiscoASA-High_CPU** – The CPU utilization level on a Cisco ASA firewall is high. • **Fortinet-Firewall_Update_Failed** – An update performed by a Fortinet firewall has failed. • **Exim-Empty_SMTP_Message** – The Exim email server encountered an email that was empty. • **Linux-User_Exists** – An attempt to add a user failed because the user account name already exists. • **Windows-Print_Warning** – An attempt to print a document from Windows encountered an issue.

file-access	Denotes any type of sniffed PVS network session or log that indicates that a file was accessed, modified or likely retrieved. Examples: • **FTP-File_Upload** – An FTP server logged a file being uploaded. • **FTP-File_Renamed** – An FTP server logged a file being renamed. • **PVS-SMB_Client_DLL_File_Download** – The PVS sniffed a file with a `.dll` extension being downloaded through Windows file sharing. • **PVS-Web_File_7Z_Request** – The PVS sniffed a file with a `.7z` extension downloaded over HTTP. • **PVS-Web_Executable_RPM_Request** – The PVS sniffed a file with a `.rpm` extension downloaded over HTTP.
firewall	Denotes any type of log from a firewall, an intrusion prevention device, a router or a firewall or application configured at the local host to specifically deny connections. Logs from a firewall about an incorrect configuration, administrator logins, port scan detection or errors would be normalized to other event types. Some Web Application Firewalls (WAFs) have their events normalized to the `web-error` event type and not the `firewall` event type. Examples: • **Checkpoint-Blocked_TCP** – A Checkpoint firewall blocked a TCP connection. • **CiscoASA-Blocked_ICMP** – A Cisco ASA firewall blocked an ICMP packet. • **Fortinet-Firewall_Virus_Oversized** – An email with an attachment larger than a size set by policy was blocked by a Fortinet firewall. Fortinet firewall messages that indicate virus activity are normalized to the `virus` LCE event type. • **Microsoft_Drop_TCP** – A local Microsoft firewall denied a TCP connection. • **TippingPoint-Block_UDP_Critical** – The HP TippingPoint IPD blocked a UDP network session.
honeypot	Indicates logs that are normalized from applications designed to simulate networks, hosts and applications for the purpose of detecting intruders. Examples: • **Nepenthes-Critical_Alert** – A critical log from the Nepenthese malware research toolkit has occurred. • **Honeyd-TCP_Connection_Reset** – The `honeyd` application reset a TCP connection. • **Forescout-User_Mark** – The ForeScout product has marked a user and will track if they return to the network.
Indicator	The "indicator" event type is used by LCE to track correlations associated with scanning, compromises, anomalies, and other behaviors that indicate the presence of determined attackers, advanced malware, and other forms of potentially malicious activities. The tracking occurs for any IP address that has had at least two unique events occurring in a short period of time. The alert level is increased for each new type of event added into the tracking of the IP address. Each time an alert level is increased, a new log is produced summarizing the sequence of events. LCE tracks various different normalized event types that are shown in the “Indicator” type, including large or medium anomalies. Also included is the threatlist (botnet) inbound, and outbound activity that indicates file transfers or proxy traffic. The “Indicator” type also tracks IDS events, continuous events, changes (network, account), downloads, and other behaviors

	of interest. When “Indicator” is selected, events will be displayed that look similar to the example below: > Indicator_Alert-Level_02 – The “Indicator” value will increase with each event associated with that particular “Indicator” up to the value of 20. The number at the end of an “Indicator Alert” is associated with the number of events that occurred related to one IP address. Drilling down further in the event will reveal the list of events. An example of an “Indicator Alert” with a level of 5 is shown below. `Indicator_Alert level 5 for IP 111.222.333.444 with 5 events: malicious executable detected (PVS-Potential_Serving_of_Malicious_EXE), followed by single system scanning activity (Intrusion_Network_Scan), followed by detection of cleartext user authentication (PVS-User_Authentication_Detected), followed by multiple web server scanning activity (Web_Servers_Scanned), followed by ongoing web scanning activity (Long_Term_Web_Error_Activity`
intrusion	Denotes logs from network IDS, firewall, application and operating systems that indicate some sort of network attack. Post scans, denial of service and logs that indicate virus probes are normalized to their own LCE event types. Examples: • **Bind-Potential_Attack** – The BIND DNS server processed a DNS request that indicated a known attack type. • **CiscoPIX-Potential_SNMP_Overflow_Attempt** – A Cisco PIX firewall encountered an SNMP query that was likely a buffer overflow attack. • **Fortinet-TCP_IDS_Event** – The Fortinet firewall encountered a generic IDS event occurring over TCP. • **IMAP-User_Overflow** – The WU IMAP server encountered a very long user name that likely indicates a buffer overflow attempt. • **Bro-SMTP_Event** – The Bro IDS encountered an email-based attack. • **Dragon-Compromise_Event** – The Enterasys Dragon IDS encountered a compromise attack attempt. • **Intrushield-Buffer_Overflow** – The McAfee Intrushield IPS encountered a buffer overflow attempt. • **Snort-TCP_Attempted_Information_Leak** – The Snort IDS encountered an event classified as an Attempted Information Leak attempt.
lce	The LCE includes this distinct event type to assist in tracking information about LCE clients such as the LCE Windows client, LCE Linux client, LCE NetFlow Client (TFM) and the LCE Network Client (TNM). The LCE clients generically log heartbeats and system usage. Examples: • **LCE-Client_Login** – An LCE client has logged into the LCE. • **LCE-High_Load** – Based on the configuration of the **`system_monitor.tasl`** script, CPU loads above a certain value can cause this alert. • **Windows-LCE_Client_Disk_Space** – Report of current disk space available on a Windows system. • **Windows-LCE_Client_Deleted_File_On_Size** – The LCE client deleted a watched log file because it exceeded the configured maximum allowed size.

login	Indicates any type of login event to an application, operating system, VPN, firewall or other type of device. Examples: • **CiscoASA-Admin_Permited_Console** – A Cisco ASA had an administrator successfully authenticate via the physical console interface. • **FTP-Valid_Login** – An FTP server had a valid user authenticate. • **IMAP-User_Login** – An IMAP server had a user successfully authenticate to receive their email. • **Unix-SU_Event** – A Unix system had a user use the super user "su" command. • **Windows-Successful_Network_Login** – A Windows system logged in a user from the network.
login-failure	Denotes any type of authentication log that indicates credentials were presented and were incorrect. This is distinct from application logs that block an IP address or access to resources that were denied. These log as event types of **`firewall`** or **`access-denied`** respectively. Examples: • **DLink-Admin_Login_Failure** – A D-Link home router had a login failure for the administrator account. • **Filezilla-Incorrect_Password** – The FileZilla FTP server encountered a login failure for a user account. • **Password_Guessing** – The LCE has correlated multiple password login failure events. • **Successful_Password_Guess** – The LCE has observed multiple Password Guessing events followed by a successful login. • **Cisco-NAC_Invalid_Login** – A Cisco NAC appliance has had a user unable to authenticate. • **Unix-Su_To_Root_Failed** – A Unix system had a user account attempt to run the "su" command but was unable to provide the proper credentials. • **Windows-Logon_Failure** – A Windows system had a user unable to authenticate.
logout	The LCE normalizes events for applications, operating systems and devices that detect when a user's session is finished to the **`logout`** event type. Examples: • **CiscoASA-SSH_Disconnect** – A Cisco ASA firewall had an established SSH session finish. • **WatchGuard-VPN_User_Logged_Out** – A VPN user of a Watchguard firewall has finished their session. • **SC4-Logout** – A SecurityCenter user has logged out of the application.
nbs	The LCE tracks all normalized events that have occurred for each host. As new normalized events are logged for the host, the LCE will generate secondary events based on the event type. For example, perhaps a Linux server is set up with SSH and has only ever had users login with passwords. The first time a user tried to login with a certificate, the resulting SSH-Failed_Publickey event would cause the LCE to generate a Never_Before_Seen-Login-Failure_Event. Examples: • **Never_Before_Seen-Access_Denied** – The host generated a normalized **`access-`**

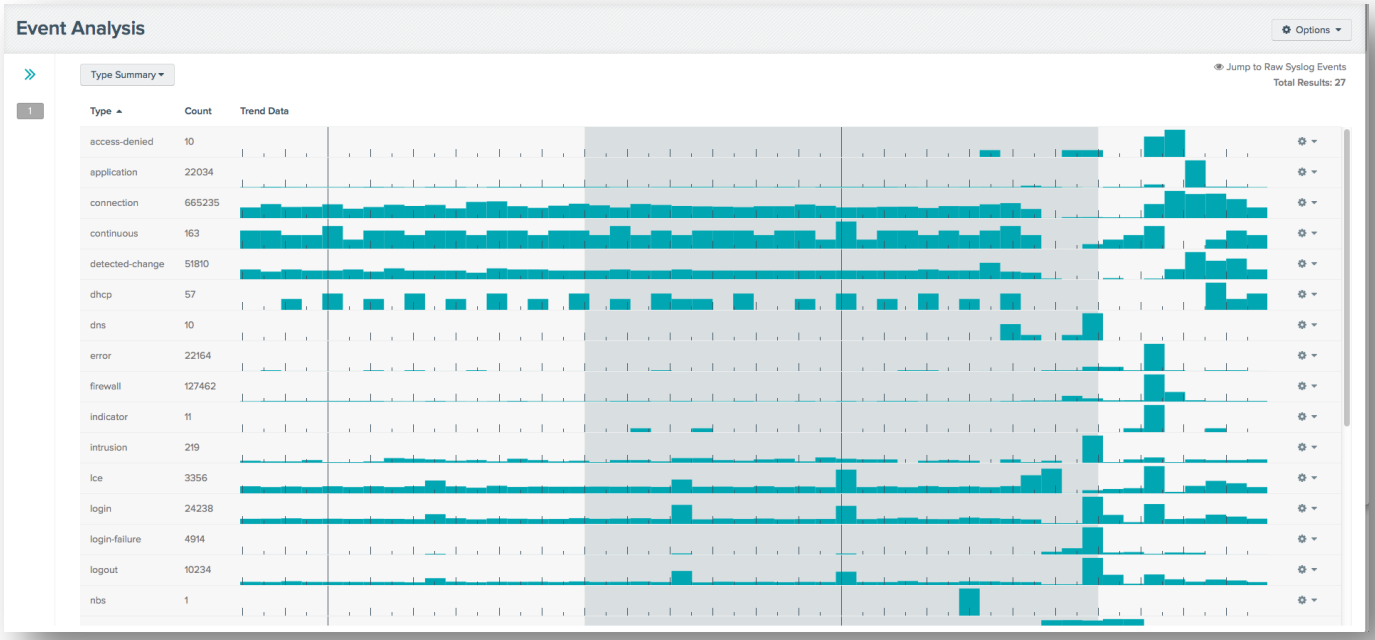
	`denied` event log that had never been seen prior to this log. • **Never_Before_Seen-Firewall** – The host generated a normalized `firewall` event log that had never been seen prior to this log. • **Never_Before_Seen-System** – The host generated a normalized `system` event log that had never been seen prior to this log.
network	Logs from the Tenable NetFlow Montior (TFM) and the Tenable Network Monitor (TNM) are logged to this LCE event type. Event names are used to designate the collection type (TNM or TFM) as well as session length and amount of bandwidth transferred. Logs from the PVS, ArpWatch and some other sources that indicate network changes are also logged. Examples: • **TFM-Long_TCP_Session_15_Minutes** – A Netflow session lasting about 15 minutes was observed. • **TFM-Long_TCP_Session_Many_Hours** – A Netflow session lasting several hours was observed. • **TFM-TCP_Session_Whole_1-10MB** – A sniffed session that transferred between 1 and 10 MB of data was observed. • **PVS-SMTP_Proxy** – The PVS has observed a host proxy SMTP emails. • **Suspicious_Proxy** – A host was generically observed to have multiple network connections that could be indicative of a proxy.
process	Logs from Unix process accounting and Windows event logs that indicate process starts and stops, as well as executable crashes, restarts, hung states and segmentation faults are logged to this LCE event type. The LCE will also summarize a variety of user and process events as well as child-parent relations for executables which get logged here as well. Examples: • **Bind-Fatal_Exit** – The DNS BIND application had a fatal crash. • **New_Unix_Parent_Child_Pair** – The LCE has detected a new parent and child pair that indicate an existing process was invoked by a new parent. This could indicate a hacked system or an administrator using the system in a new way. • **FreeBSD-Root_Command_Issued** – A FreeBSD system had a command issued by root. • **Hourly_Hung_Summary** – The LCE generated an hourly list of programs that have hung for a given system. • **Windows-New_Process_Created** – A Windows system logged the start of a new process. • **New_Command** – The LCE has detected a Unix or Windows command that was never previously run on a system.
restart	The LCE will normalize logs from when applications, services, router, switches, devices and operating systems reboot, restart and are shutdown to the `restart` event type. Applications that have crashes, core dumps and other types of specific process related issues are logged to the `process` event type. Examples: • **Unix-Syslog_Restarted** – The Unix syslog process has restarted. • **Windows-Unexpected_Shutdown** – A Windows system has logged a shutdown that was unexpected. • **MSSQLSVR-Pause_Request** – A MS SQL server received a request to pause operation.

	• **Nessus-Restarting** – The Nessus vulnerability scanner is restarting.
scanning	Network IDS, firewall, antivirus and other log sources that detect port scans, port sweeps and probes are logged to the LCE **`scanning`** event type. Examples: • **Bro-PortScan** – The Bro IDS has discovered a port scan event. • **NetScreenIDP-Port_Scan_UDP** – The NetScreen IDP has discovered a UDP port scan. • **SnortET-Scanning** – A Snort Emerging Threats rule has alerted on a type of portscanning activity. • **OSX-Limiting_RST_Response** – A Mac OS X encountered many network connections, which is typically the result of port scanning. • **PVS-New_Host_Portscanning** – The LCE has detected a brand new host on the network that immediately started to perform scans of some sort.
social-networks	The PVS detects a wide variety of social network activity such as Bing searches, logins to Gmail, Facebook, Wikipedia searches, Twitter and generic passively discovered IMAP and POP access. These are logged to the **`social-networks`** LCE event type. Examples: • **PVS-Web_Query_Yahoo_Search** – PVS has logged a query to the Yahoo search engine. • **PVS-FTP_UserID_Enumeration** – PVS has logged an FTP access and saved the account name. • **PVS-MSN_Messenger_Login_Detection** – PVS has observed a login to the Microsoft Messenger service and logged the username. • **PVS-Facebook_Usage_Detection** – PVS has observed a login to Facebook.
spam	Logs from email servers, antivirus email tools, spam appliances, firewalls and other sources that indicate spam activity are normalized to the LCE **`spam`** event type. Examples: • **Postfix-SPF_Mail_Rejected** – The Postfix email server rejected an email. • **StealthWatch-High_Volume_Email** – The Lancope Stealthwatch NBAD product detected a high volume of SMTP messages. • **Amavis-Blocked_Spam** – The Amavis spam tool detected a spam message and dropped it.
stats	For every unique type of event, the LCE will profile the frequency of events and alert when there is a statistical deviation for any event. These events are normalized to the **`stats`** LCE event type. For example, for a given host, if the LCE detected a large spike in the frequency of the PVS-Web_Query_Yahoo_Search event, it would issue a Statistics-Social_Networks_Large_Anomaly event. Examples: • **Statistics-USB_Large_Anomaly** – The LCE detected a large anomaly in USB (event type **`usb`**) insert or removal events. • **Statistics-Web_Access_Medium_Anomaly** – The LCE detected a medium anomaly in the amount of web-access event types for a given host.

system	The LCE will normalize operating system, router, switch or device logs of significance to the event type of **`system`**. Login failures, errors and application events are logged to other event types. Examples: • **Fortinet-Firewall_Added_Local_User** – A Fortinet firewall had a new local user added to it. • **Promiscuous_Mode_Enabled** – Sniffing was enabled on a Unix system. • **Linux-Group_Removed** – A Linux system had a group removed. • **Windows-Firewall_Change** – A Windows server had a firewall rule changed. • **Windows-Directory_Service_Created** – A Windows server had a directory service created. • **Windows-Registry_Changed** – A Windows system had a registry value changed.
threatlist	The LCE maintains a list of hostile IPv4 addresses that are known to be participating in botnets. The LCE considers connection and network events to detect when a hostile IP address connects inbound to your network as well as when a host on your network connects outbound. These events are normalized to the **`threatlist`** event type. Examples: • **Threatlist_Inbound_Connection_SSH** – A known hostile host has connected to a system on your network via SSH • **Threatlist_Outbound_Connection_HTTPS** – A local host has connected on port 443 to a known hostile IP address. • **Threatlist_Outbound_Connection_High_Port** – A local host has connected on a port larger than 1024 to a known hostile IP address.
usb	The LCE windows client can detect USB and CD-ROM insertions and removals. The logs generated by these events are normalized to the **`usb`** event type. Examples: • **Windows-LCE_Client_Detected_Attached_Drive** – A USB or CD-ROM drive was attached. • **Windows-LCE_Client_Detected_Removed_Drive** – A USB or CD-ROM drive was removed. • **Windows-LCE_Client_Detected_Attached_USB_Device** – A USB peripheral was attached. • **Windows-LCE_Client_Detected_Removed_USB_Device** – A USB peripheral was removed.
virus	Logs that indicate the presence of a virus in email, a virus found on a system by an antivirus agent, virus logs found by network IDS events and firewalls are normalized to the LCE event type of **`virus`**. Information about the status of an antivirus agent, such as the agent reporting a successful update of antivirus signatures, are sent to the event type of **`application`**. Similarly, activity from a virus infection could be normalized to **`intrusion`** events, **`threatlist`** connections and **`scanning`** events. Examples: • **Symantec-Virus_Warning** – A Symantec antivirus agent found a virus. • **Sophos-Email_Quarantined** – A Sophos email processing component found a virus present in an email and quarantined it.

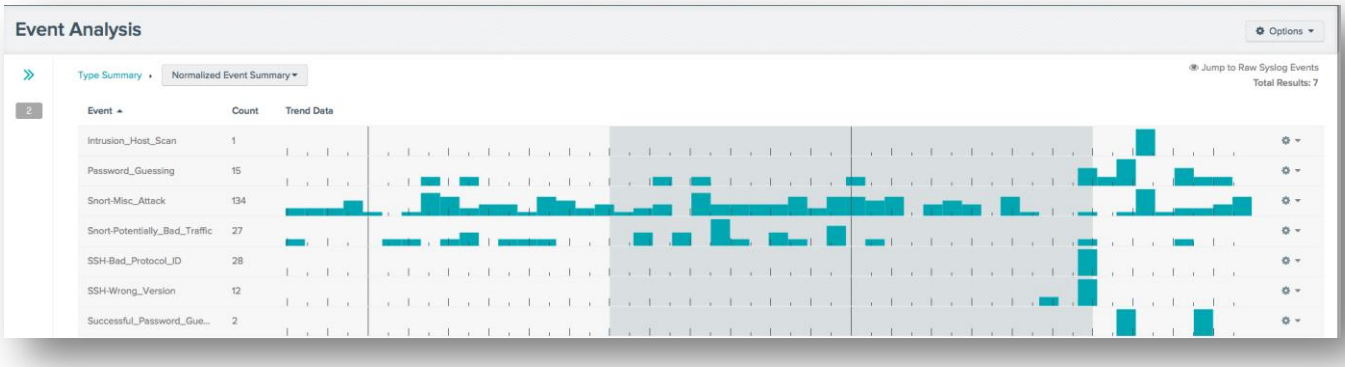
	• **McAfee-Warn_Mode_Would_Be_Blocked** – A McAfee antivirus agent found a virus and would have blocked activity associated with it but the agent is in warning mode. • **SnortET-Malware_Activity** – A Snort IDS sensor running the Emerging Threats ruleset has detected malware activity.
vulnerability	As security issues and new information about systems and networks are reported as part of the vulnerability monitoring process, the LCE normalizes these event types to the **`vulnerability`** category. Some new types of information such as new port browsing events and new hosts are reported to the **`detected-change`** event type. The primary source of these logs is the PVS. Examples: • **Intrushield-Unwanted_Software** – A McAfee IntruShield IPS has found software that is likely unwanted. • **RNA-OS_Confidence_Update** – A SourceFire RNA sensor has observed enough traffic to update its guess for the operating system of a given node. • **PVS-High_Vulnerability** – A high severity vulnerability was passively discovered on a given node.
web-access	Any type of log that indicates a successful connection to a web resource is normalized as a **`web-access`** LCE event type. Logs gathered by web servers, web proxies, firewalls and load balancers that indicate connections to web services are logged here. Note that **`web-access`** events can refer to a host on the Internet connecting to a public web server or internal users accessing the Internet through a web proxy. Another noteworthy type to consider is the **`file-access`** type. Event types log specific files obtained over HTTP. As improvements are made to parsing of web logs, breaking them out by file extension and moving events to the file-access category will occur. Examples: • **CiscoASA-Accessed_URL** – A Cisco ASA firewall logged a connection to a web site and logged the specific URL. • **Apache-Valid_Web_GET_Request** – An Apache web server logged a valid GET request to a resource hosted by it. • **IIS-Move_Request** – An IIS web server encountered a request to move a resource. • **Squid-TCP_Miss** – A Squid web proxy encountered a valid request for a web site, but the web site was not present in the existing cache.
web-error	Denotes any type of web access event that is denied because the file does not exist, the server responded with an error or a firewall or web application firewall blocked the access. These logs are generated every day by users who reference incorrect URLs, but are also generated during web application probes. Examples: • **Apache-Invalid_Method** – An Apache web server encountered a request that referenced an invalid HTTP method. • **IIS-Bad_Get** – An IIS web server logged a GET request that returned an error. • **Web_GET_Forbidden** – A generic HTTP GET request was deemed forbidden and logged as an error. • **Squid-Proxy_Denied** – The Squid web proxy log denied a web proxy request. • **Web_POST_ServerError** – A generic attempt to perform an HTTP POST to a web server encountered an error.

Below is an example screen capture of events as shown in a SecurityCenter for a 24 hour period:



This portion of events (the screen image only shows “access-denied” types through “intrusion” types for 24 hours) offers some interesting patterns. The dark areas represent 6:00 PM through 6:00 AM. The activity graph shows that some types of events occur all the time, such as the “intrusion” types and “error” types. If a SecurityCenter user were to mouse over the graphs, they would be shown a pop up image of the total amount of events at that point and the exact times.

Clicking on any of the event counts would bring the user to a list of unique events. For example, clicking on the “219” next to the “intrusion” type would show:



This similar display displays the specific different set of normalized events that are part of the “intrusion” type.

Event Schema

Every normalized event will attempt to populate as much of the following schema as possible:

- Source and destination IP addresses
- IP Protocol
- Source and Destination Ports
- Unique Event Name
- Unique Event Type
- Sensor name
- Time of Event
- User Name

For example, the above “promiscuous” mode message is parsed by the LCE rule number 1316. The rule is shown below:

```
id=1316
name=Promiscuous mode enabled.
match= kernel:
match=: Promiscuous mode enabled.
log=event:Linux-Promiscuous_Mode_Enabled type:system
```

If this rule matches an event, it will populate the schema with the source IP of either the original **`syslog`** message or the LCE client. It will also set an event name of “Linux-Promiscuous_Mode_Enabled” and an event type of “system”.

This is an extremely simple normalization rule. This document will provide several examples that normalize similar log messages as well as more complex log messages such as firewall and intrusion detection logs.

Libraries

The LCE daemon references the **`.prm`** libraries located in the **`/opt/lce/daemons/plugins`** directory.

Each library contains one or more LCE plugins that are designed to parse specific types of log files. The libraries may also contain specific information about the devices they support. For example, the library that parses certain types of Windows security events may contain information on how the Windows LCE Client should be configured.

Unnormalized Events

It is quite likely that the LCE will receive logs for events but not have a rule designed to parse those events. The LCE is designed to only normalize what it has been told how to normalize.

When LCE receives logs that do not have a rule, it will categorize those events as “unnormalized”. Through analysis of the contents of the unnormalized events, users can see what the actual log messages look like and can modify their **`.prm`** libraries accordingly.

Matching Multiple Events

The LCE can also be configured (via the LCE GUI in the Advanced menu under Debugging) to continue to parse each log message across the entire **`.prm`** library of plugins. This is useful if more than one plugin is destined for a specific type of log file.

For example, for a firewall that performs network address translation, each log may have three IP addresses. One IP address will be the internal address, another would be the public address and the third would be an Internet address. A

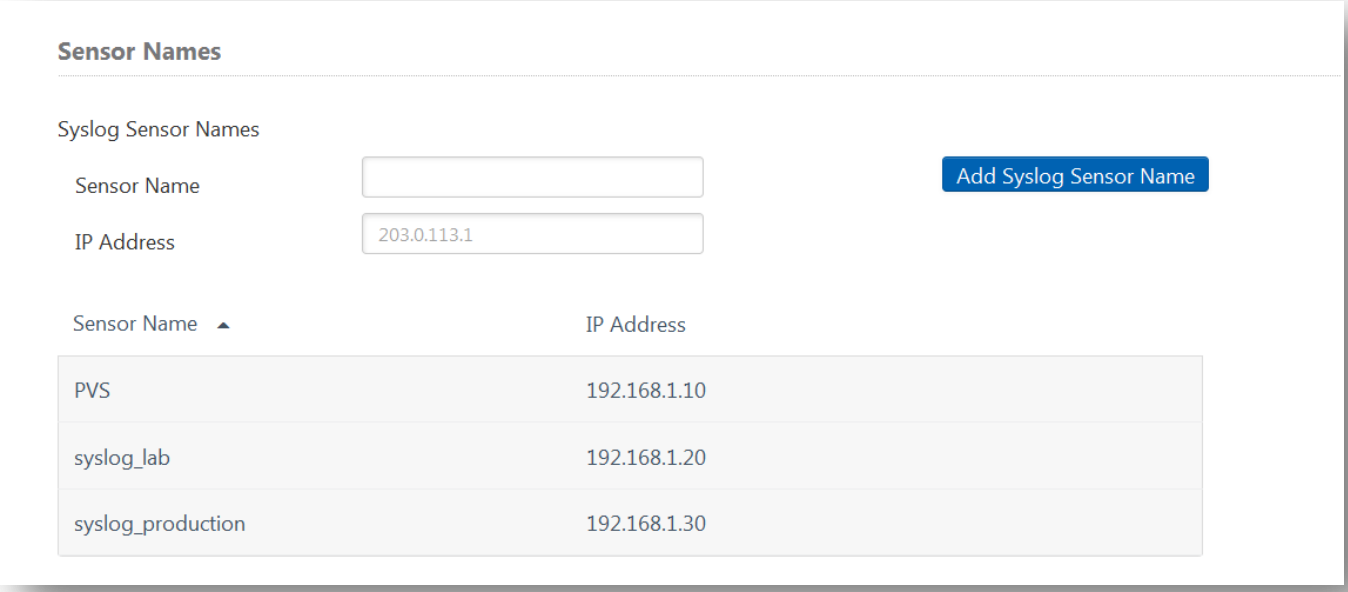
university may have their students on a private network such as 10.10.22.123, translate them to a public university IP address and finally log the fact that they downloaded their email from Google's web site.

By allowing the LCE to parse the same log multiple times, multiple events can be logged. More examples for the reason why multiple log hits would be required could be:

- Separate rules to parse web logs for valid/invalid web hits as well as performing IDS analysis of the URLs for suspicious attacks
- Separate rules for generic firewall accept/deny as well as matching on specific denied ports
- Separate rules for valid/invalid logins as well as alerting on specific user names such as "root" and "administrator"

Force Sensor Name for Events

In many cases, by the time **`syslog`** data reaches the LCE, the true source of the UDP packet is lost because of packet modification by intermediary forwarding systems. Because of this, LCE may inadvertently normalize data with a sensor name that does not truly match the originating system. To remediate this, a sensor named can be assigned to each IP address sending data to the LCE. This sensor name will be associated with all logs from the designated source, regardless of whether or not another sensor name is extracted from the log text. The configuration is located in the LCE GUI under Configuration, Advanced, Sensor Names. An example is shown below:



Writing Log Correlation Engine Plugins

Log Correlation Engine Database Events

Tenable's LCE is designed to receive log messages, parse out relevant fields of those log messages, and then store the results in a proprietary database. Each entry stored in the database is known as a LCE event and contains the following fields:

- **srcip** – the source IP address of the event
- **dstip** – the destination IP address of the event

- **srcport** – the source port of the event
- **dstport** – the destination port of the event
- **proto** – the protocol of the event
- **sensor** – the name of the sensor that sent the event
- **event** – the actual event name
- **type** – the type of sensor that sent the event (IDS, firewall, etc.)
- **message** – the full log message that triggered the storing of this event
- **user name** – the actual login name used for the matching user

An event stored in the database may not have all of these fields set. The number of the fields that are set depends on how much information was parsed and recorded by the LCE when receiving each log message. At a minimum, the event name and message fields should always be set.

Log Correlation Engine Plugin Keywords

There are several keywords available for writing plugins for the LCE. Some of these keywords are mandatory and some are optional.

Keyword	Description
name=<plugin name>	This keyword specifies the name of the plugin. The name of the plugin generally describes what sort of log event this plugin is designed to trigger on. Multiple LCE plugins may have the same name.

match=<match>	Match strings have the following format:
	
	`[!]string`
	
	! : Means that the packet must NOT match <string>
	
	A plugin may have multiple match statements.
	
	Here is a full example for a Snort ICMP event:
	
	`match=snort:`
	`match=->`
	`match={ICMP}`

regex=<regex>	This keyword specifies a complex regular expression search rule that will be applied to the log message. A regex is applied only after a log message has matched all of the "**match=**" patterns. For example:
	
	`match=snort:`
	`match=->`
	`match={ICMP}`
	`regex=snort:.+\] ([a-zA-Z0-9_ \.]+) \[.*\{ICMP\} ([0-9]+(\.[0-9]+){3}) -> ([0-9]+(\.[0-9]+){3})`
	
	Parentheses are used in this regular expression to allow the LCE to remember and save matched patterns in a received log message. These matched patterns are

	generally the fields of the log message that you want to save in an event structure.
	
	If the above regex was applied against the following Snort ICMP alert, then the following fields of the log message would be saved because of the presence of parenthesis in the regex rule:
	
	syslog alert:
	
	`<33>Mar 24 08:01:33 fwall snort: [1:483:2] ICMP PING CyberKit 2.2 Windows [Classification: Misc activity] [Priority: 3]: {ICMP} 192.168.0.1 -> 192.168.0.2`
	
	Saved patterns:
	
	`"ICMP PING CyberKit 2.2 Windows", "192.168.0.1", "192.168.0.2"`
	
	These matched patterns may be recalled using dollar sign notation to indicate which match we are referring to. These are of the forms $[0-9] where the digit 0-9 specifies which pattern (in the order each pattern was matched) to recall. So for the above example, each pattern could be recalled resulting in these matches:
	
	`$1 - which recalls "ICMP PING CyberKit 2.2 Windows"`
	`$2 - which recalls "192.168.0.1"`
	`$4 - which recalls "192.168.0.2"`
	
	Notice that the destination IP address 192.168.0.2 is recalled using $4 and not $3. This is because each IP address regex pattern actually contains two sets of parenthesis:
	
	`([0-9]+(\.[0-9]+){3})`
	
	The first set of parenthesis, which are the outer parenthesis, match on the IP address as a whole. The second set of parenthesis, which are the inner parenthesis match on the last three octets of the IP address. Thus, this IP address regular expression results in two saved patterns. It just happens that in this case, we are only interested in the first of these saved patterns (the IP address as a whole).
	
	  A match statement OR a regex is required. A plugin may contain multiple match statements but only one regex statement.

log=<log options>	The "**log**" keyword provides a way to specify what information to store in the LCE database if this plugin matches a received log message. As stated earlier, each entry stored in the LCE database is stored as a LCE event structure. The **log** keyword enables one to set each field of a stored event to either a value extracted from the received log message (using the saved patterns from the regex rule) or to a hard coded value written explicitly in the log statement. In this way a person can save only those pieces of a log message that are useful. For example, the regex rule may specify a search rule for an IP address pattern. If this search rule matches an IP address in a received log message, then the **log** keyword allows one to specify whether the IP address should map to the source IP or the destination IP when storing the event for this message.

Mapping may be done for each field in an event structure using the following keywords: **srcip**, **dstip**, **proto**, **event**, **type,user** and **sensor**. The syntax for each mapping is as follows:

```
keyword:<value>
```

Where keyword is one of **srcip**, **dstip**, **proto**, **event**, **type**, **user** or **sensor** and <value> is either written explicitly or is a reference to a saved match pattern from the regex rule.

For example, suppose we had the following set of match patterns and regex rules designed to match a Snort ICMP event:

```
match=snort:
match=->
match={ICMP}
regex=snort:.+\] ([a-zA-Z0-9_ \.]+) \[.*\{ICMP\} ([0-
9]+(\.[0-9]+){3}) -> ([0-9]+(\.[0-9]+){3})
```

An example Snort event that we would match on in this case is:

```
<33>Mar 24 08:01:33 fwall snort: [1:483:2] ICMP PING
CyberKit 2.2 Windows [Classification: Misc activity]
[Priority: 3]: {ICMP} 192.168.0.1 -> 192.168.0.2
```

Here the regex contains parenthesis so as to save the event name and source and destination IP addresses from this field. In order to store this event to the LCE database, we could set up the following log options:

```
log=srcip:$2 dstip:$4 proto:1 event:Snort_ICMP_event
type:ids
```

Here the event name and protocol are hard-coded and the rest of fields are based on extracted patterns from the received log message as specified by the regex rule. Note that if a value is hard coded it may NOT contain spaces. So in the above log rule, "**event:Snort_ICMP_event**" could NOT have been written as "**event:Snort ICMP event**".

Only the event mapping must be specified when writing a plugin. The rest of the mappings are optional. If no **dstip** mapping is specified, then by default the LCE will set the destination IP address of an event to the address of the machine it received the log message from. All other mappings are zero-valued if not specified in the log rule.

The **log** keyword is always required and within the log statement, the **event** keyword (e.g., "**event:snort_tcp_event**") is required as well.

User Tracking

The LCE can be used to track the IP addresses a network user is originating from and then tag any other events from that IP address as belonging to that user.

To make use of this, the LCE needs to know which types of logs in your organization contain the username you wish to correlate and track IP addresses from. Please refer to the "LCE Administration and User Guide" to learn how to configure this. When the LCE is configured, it will only use PRM IDs that have been configured to process user to IP address relationships.

Consider the following PRM from the **auth_bluesocket.prm** file:

```
id=1032
name=The BlueSocket device had a network user start their network session.
match=user_tracking:
match=event=user_l
match=event=user_login_successful
match=obj=user
regex=&ipaddr=([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)&name=(.*)&msg=
log= type:login event:BlueSocket-User_Login srcip:$1 user:$2
```

An actual user authentication message for BlueSocket looks like this:

```
<181>user_tracking:
       event=user_login_successful&loglevel=notice&obj=user&ipaddr=192.168.0.1&name=us
       er1@nessus.org&msg=Login RADIUS user user1@nessus.org on Primary RADIUS server
       at [00:13:02:89:d1:f5]/192.168.0.1 as role Authenticated, login time = 2008-06-
       17 20:39:29, sessionID = 00:0E:0C:73:A4:E0:121374956977878&
```

The “regex” and “log” statements in the PRM extracts the string 192.168.0.1 and assigns it to srcip, and the username is assigned “user1@nessus.org”.

If the LCE has not been configured to process ID 1032 as a source of user to IP address tracking, the event is normalized and logged as any other event. However, if ID 1032 were enabled for user to IP tracking, the LCE would perform the following steps:

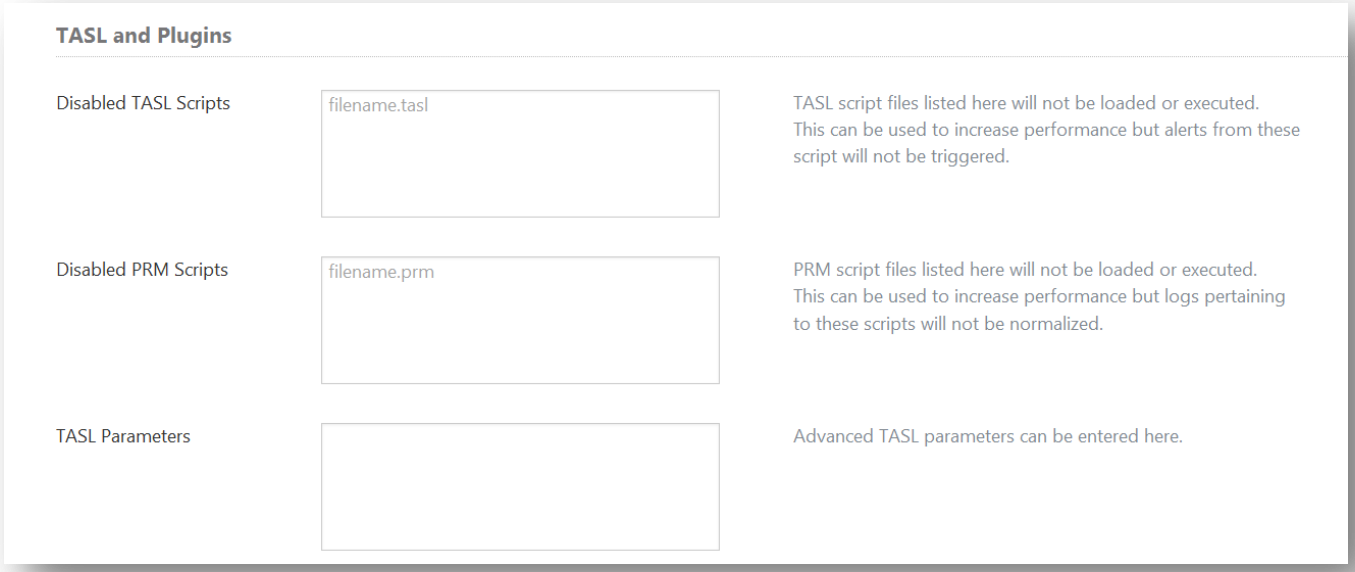
- It would check the current table of usernames. If this were a new user, an additional “new user” event would be generated.
- It would also check the current table of usernames and IP address matches. If this were a new IP address for an existing user, then a new user to IP address message would be generated.
- Lastly, for any other events originating from the IP address of this login (in this case 192.168.0.1) they would all be automatically assigned a username of “user1@nessus.org”.

LCE login-failure plugins do not normalize usernames because those logs are not assured to provide a valid username, and it would contaminate the username database. Additionally, it is advised never to add a login-failure plugin ID into the list of User Tracking Plugins. Doing so would invalidate user tracking for hosts that triggered the plugin.

Log Correlation Engine Plugin Management

The LCE is distributed with a large number of plugins, but they may not all be of interest in a particular environment. To improve performance and facilitate log analysis, configure the LCE server with only the plugins desired for analysis. To disable a PRM go to the LCE GUI, select the “Configuration”,” Advanced” menu, and then navigate to the “TASL and Plugins” section. Add in the full name of the PRM you wish to be ignored and then select “Update” at the bottom of the “Advanced” menu.

In the same manner, to disable a TASL script, navigate to the “TASL and Plugins” section and add the full name of the TASL you wish to be ignored and then select “Update” at the bottom of the “Advanced” menu. (TASL scripts are covered further in the LCE TASL Reference Guide documentation).



Desired libraries, modified libraries and custom plugins should be placed in **`/opt/lce/daemons/plugins`** directory and given a plugin ID of 25000-27999.

Modifying existing Tenable PRM files is not recommended, as any update process will overwrite them with subsequent changes made by Tenable.

Log Correlation Engine Clients

There are a number of LCE clients available for a variety of events and operating systems. Each of the available LCE clients are covered in a separate document entitled “LCE Client Guide”.

For More Information

Tenable has produced a variety of documents detailing the LCE's deployment, configuration, user operation, and overall testing. These documents are listed here:

- Log Correlation Engine 4.2 Architecture Guide – provides a high-level view of LCE architecture and supported platforms/environments.
- Log Correlation Engine 4.4 Administrator and User Guide – describes installation, configuration, and operation of the LCE.
- Log Correlation Engine 4.4 Quick Start Guide – provides basic instructions to quickly install and configure an LCE server. A more detailed description of configuration and management of an LCE server is provided in the "LCE Administration and User Guide" document.
- Log Correlation Engine 4.4 Client Guide – how to configure, operate, and manage the various Linux, Unix, Windows, NetFlow, OPSEC, and other clients.
- LCE 4.4 High Availability Large Scale Deployment Guide – details various configuration methods, architecture examples, and hardware specifications for performance and high availability of large scale deployments of Tenable's Log Correlation Engine (LCE).
- LCE Best Practices – Learn how to best leverage the Log Correlation Engine in your enterprise.
- Tenable Event Correlation – outlines various methods of event correlation provided by Tenable products and describes the type of information leveraged by the correlation, and how this can be used to monitor security and compliance on enterprise networks.
- Tenable Products Plugin Families – provides a description and summary of the plugin families for Nessus, Log Correlation Engine, and the Passive Vulnerability Scanner.
- Log Correlation Engine Log Normalization Guide – explanation of the LCE's log parsing syntax with extensive examples of log parsing and manipulating the LCE's `.prm` libraries.
- Log Correlation Engine 4.4 TASL Reference Guide – explanation of the Tenable Application Scripting Language with extensive examples of a variety of correlation rules.
- Log Correlation Engine 4.0 Statistics Daemon Guide – configuration, operation, and theory of the LCE's statistic daemon used to discover behavioral anomalies.
- Log Correlation Engine 3.6 Large Disk Array Install Guide – configuration, operation, and theory for using the LCE in large disk array environments.
- Example Custom LCE Log Parsing - Minecraft Server Logs – describes how to create a custom log parser using Minecraft as an example.

Documentation is also available for Nessus, the Passive Vulnerability Scanner, and SecurityCenter through the Tenable Support Portal located at https://support.tenable.com/.

There are also some relevant postings at Tenable's blog located at http://www.tenable.com/blog and at the Tenable Discussion Forums located at https://discussions.nessus.org/community/lce.

For further information, please contact Tenable at support@tenable.com, sales@tenable.com, or visit our web site at http://www.tenable.com/.

About Tenable Network Security

Tenable Network Security provides continuous network monitoring to identify vulnerabilities, reduce risk, and ensure compliance. Our family of products includes SecurityCenter Continuous View™, which provides the most comprehensive and integrated view of network health, and Nessus®, the global standard in detecting and assessing network data.

Tenable is relied upon by more than 24,000 organizations, including the entire U.S. Department of Defense and many of the world's largest companies and governments. We offer customers peace of mind thanks to the largest install base, the best expertise, and the ability to identify their biggest threats and enable them to respond quickly.

For more information, please visit tenable.com.