

Table of Contents

Meeting Information Security Compliance Mandates: Secret Server and QRadar Security Intelligence Platform Integration and Configuration	2
The Secret Server Approach to Privileged Account Management	2
Risks and Benefits	2
Initial Configuration and Event Log Analysis	3
Exporting Logs from Secret Server	3
Upload a Custom Parser	3
Custom Parsers	4
Example Raw Payload	4
Example Custom Parser	4
QID Mappings	5
Event List	6

Last updated: June 23, 2020

Meeting Information Security Compliance Mandates: Secret Server and QRadar Security Intelligence Platform Integration and Configuration

Using Secret Server event data with IBM's QRadar Security Intelligence Platform can give organizations deep insight into the use of privileged accounts. Examples of privileged accounts include the Windows local administrator, service or application accounts, UNIX root accounts, Cisco enable passwords and more. Used together, these tools provide secure access to privileged accounts, and provide greater visibility to meet compliance mandates and detect internal network threats.

THE SECRET SERVER APPROACH TO PRIVILEGED ACCOUNT MANAGEMENT

Many environments that have strict Information Security policies also require methods to control and monitor access to privileged accounts. Enterprises often apply security policies such as physical access restrictions to hardware, network firewalls, appropriate-use guidelines, and user account restrictions. For privileged accounts, access is more difficult to track and verify. Implementing privileged account management software such as Secret Server enables organizations to strictly control and track access.

Enterprises that implement Secret Server gain the ability to grant or deny granular access to critical systems. When access is granted, use of that access is tracked based on a wide range of events. Although alerting is a core function in Secret Server, managing real-time events on the aggregate can be cumbersome. Using QRadar to manage these real-time events allows users to build customized risk analysis into their privileged account management policies. Mitigating internal privilege account threats helps organizations meet compliance requirements like Sarbanes-Oxley Act (SOX), Payment Card Industry Data Security Standard (PCI DSS), the Health Insurance Portability and Accountability Act (HIPAA), and the Federal Information Security Management Act (FISMA).

RISKS AND BENEFITS

Unmanaged privileged accounts often enjoy unchecked access across a wide array of systems, networks, and databases. Unmitigated top-level access, in the wrong hands, can be devastating to an organization. The potential for liability is not limited to internal data and productivity loss, but can include criminal and civil penalties for unauthorized disclosure of private or regulated information.

Implementing an enterprise-level privileged account management system (Secret Server) with a real-time event management system (QRadar Security Intelligence Platform) allows

organizations to mitigate risk. Critical systems can be accessed only by pre-defined users. IT Security Auditors can track access based on the needs of the enterprise.

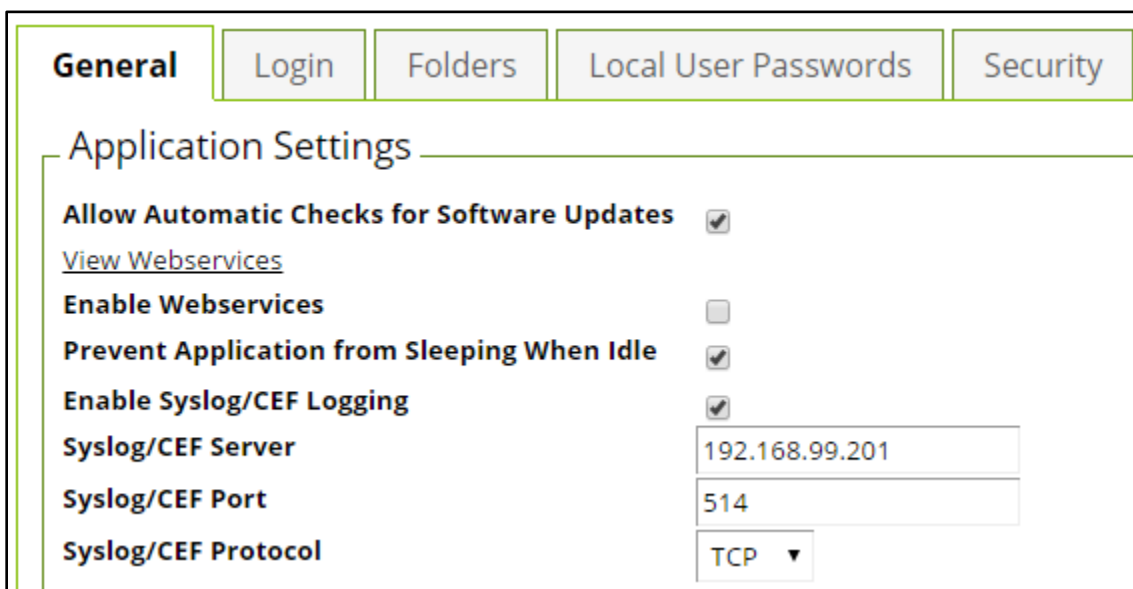
Initial Configuration and Event Log Analysis

Use the following steps to configure Secret Server and QRadar.

EXPORTING LOGS FROM SECRET SERVER

To export event logs from Secret Server to QRadar, start by logging in to Secret Server as an Administrator.

1. From the ADMIN menu, select Configuration.
2. Scroll to the bottom of the page and click Edit.
3. Select the Enable Syslog/CEF Logging check box and complete the QRadar Server IP, Port, Protocol (“TCP” for this example) before you click Save.
Data immediately begins flowing to your QRadar instance. See the following screen from the Secret Server Configuration menu:



The screenshot displays the 'General' tab of the Secret Server configuration interface. The 'Application Settings' section is expanded, showing several options with checkboxes and input fields. The 'Enable Syslog/CEF Logging' option is checked, and its corresponding fields are populated with '192.168.99.201' for the server IP, '514' for the port, and 'TCP' for the protocol.

Setting	Value
Allow Automatic Checks for Software Updates	<input checked="" type="checkbox"/>
View Webservices	View Webservices
Enable Webservices	<input type="checkbox"/>
Prevent Application from Sleeping When Idle	<input checked="" type="checkbox"/>
Enable Syslog/CEF Logging	<input checked="" type="checkbox"/>
Syslog/CEF Server	192.168.99.201
Syslog/CEF Port	514
Syslog/CEF Protocol	TCP

UPLOAD A CUSTOM PARSER

1. Log in to QRadar on your web browser.
2. Go to the Admin tab and click Log source extension.
3. Enter the appropriate fields and select the use condition as the parsing enhancement.
4. Click Choose file and select your custom-built parser, then select Upload. Apply the changes. See [Custom Parser](#) for a copy of an example parser.

5. Click Log sources and complete the field again selecting syslog as the protocol configuration.
6. Ensure your log source identifier is the host name of TSS, for example, sv-thyss
7. Select the extension that you created as your extension and make it a parsing override, then save.

Custom Parsers

1. Open the LSX_Template (Custom Parser) file in your chosen text editor.
2. Analyze the raw data that you are receiving from TSS.
3. Delete all fields from the LSX template that are not included inside TSS. For example, pre and postnat addresses.
Now you have the remaining fields that are included in TSS.
4. Enter the regular expression.

```
<pattern id="EventName"xmlns=""><![CDATA[#####]]></pattern>
```

Enter the regular expression that matches fields that you want to match into the brackets where the #'s are visible.

After you finish adding the regex for all the fields, you are done.

EXAMPLE RAW PAYLOAD

```
Aug 18 14:28:06 sv-thyss CEF:0|Thycotic Software|Secret Server|8.6.000010|18|USER -
LOGINFAILURE|2|msg=[SecretServer] Event: [User] Action: [Login Failure] By User:
domain.local\\John Snow Item Name: domain.local\\John Snow suid=6
suser=domain.local\\John Snow duser=domain.local\\John Snow duid=6
fname=domain.local\\John Snow fileType=User fileId=6 src=192.168.2.27 rt=Aug 18 2014
14:28:03
```

As you can see after the 18(Event ID) it states USER – LOGINFAILURE, which is the event name. This event is what you use to anchor the payload to QRadar. You can also map the time, username, and Source IP.

EXAMPLE CUSTOM PARSER

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Author:                Connor Hunt <connor.hunt@example.com>
Device Type:           Thycotic Secret Server
Device Version:        8.6
Protocol:              Syslog CEF
-->
```

```

<device-extension xmlns="event_parsing/device_extension">
  <!-- Do not remove the "allEventNames" value -->
  <pattern id="allEventNames" xmlns=""><![CDATA[(.*)]]></pattern>
  <!-- Everything below this line can be modified -->
  <pattern id="EventName" xmlns=""><![CDATA[\\|d+\\|(.)\\|d\\|msg]]></pattern>
  <pattern id="EventCategory" xmlns=""><![CDATA[Action: (.*)\sBy]]></pattern>
  <pattern id="SourceIp" xmlns=""><![CDATA[src=(.*)\srt]]></pattern>
  <pattern id="DestinationIp"
xmlns=""><![CDATA[(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})]]></pattern>
  <pattern id="DeviceTime"
xmlns=""><![CDATA[(\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2})]]></pattern>
  <pattern id="UserName" xmlns=""><![CDATA[By\sUser:\s(.*)\sItem]]></pattern>
  <match-group order="1" description="Log Source Extension" xmlns="">
    <matcher field="EventName" order="1" pattern-id="EventName" capture-
group="1" enable-substitutions="false"/>
    <matcher field="EventCategory" order="1" pattern-id="EventCategory"
capture-group="1"/>
    <matcher field="SourceIp" order="1" pattern-id="SourceIp" capture-
group="1" />
    <matcher field="DestinationIp" order="1" pattern-id="DestinationIp"
capture-group="1" />
    <matcher field="DeviceTime" order="1" pattern-id="DeviceTime" capture-
group="1" />
    <matcher field="UserName" order="1" pattern-id="UserName" capture-
group="1" />
    <event-match-multiple pattern-id="allEventNames" capture-group-index="1"
device-event-category="unknown" send-identity="OverrideAndAlwaysSend" />
  </match-group>
</device-extension>

```

QID Mappings

The QID or QRadar Identifier is what QRadar uses to give events their name, high-level category, and low-level category.

1. You must now create custom QIDs. Do this by SSH-ing into the QRadar console, changing the directory to /opt/QRadar/bin and running the following command:

```
./qidmap_cli.sh -c --qname <name> --qdescription <description> --severity <severity> --
lowlevelcategoryid <ID>
```

For example:

```
./qidmap_cli.sh -c --qname "USER – LOGIN" --qdescription "A user as logged in." --severity 1 – lowlevelcategoryid 19001
```

Utility option	Description
-c	Creates a new QID map entry.
--qname <name>	Type the name that you want to associate with this QID map entry. The name can be up to 255 characters in length, with no spaces.
--qdescription <description>	Type a description for this QID map entry. The description can be up to 2048 characters in length with no spaces.
--severity <severity>	Type the severity level you want to assign to this QID map entry. The valid range is 0 - 10.
--lowlevelcategoryid <ID>	Type the low-level category ID you want to assign to this QID map entry. The low-level category list is attached in the appendices.

- Alternatively, you might use a csv list as demonstrated in the Import List section and use it with the following command to import several QIDs at once:

```
/opt/QRadar/bin/qidmap_cli.sh -i -f <filename.txt>
```

19001 is used for most of the low-level category IDs as an example.

- Using the program sendnow, send the list of all events to your QRadar box (in the file called tss events all.txt) to generate every possible event. The events are described in the [Event List](#) section.
- Find the events that are received and map the events in QRadar so that they match what they are called.

Congratulations, you have successfully mapped all the TSS events!

EVENT LIST

CONFIGURATION - EDIT	The main Secret Server configuration has been edited	10	19001
FOLDER - CREATE	A Folder has been created	2	19001
FOLDER - DELETE	A Folder has been deleted	5	19001
FOLDER - EDIT PERMISSIONS	The configuration has been edited	10	19001
FOLDER - SECRET POLICY CHANGE	The policy assigned to a folder has been changed	6	19001
FOLDER - SECRET POLICY CHANGE	The Secret policy assigned to a folder has been changed	8	19001
GROUP - OWNERS MODIFIED	The owners of a group have been modified	5	19001
LICENSE - EXPIRES 30 DAYS	Secret servers license will expire in 30 days	1	19001
POWERSHELL SCRIPT - CREATE	A PowerShell script has been created	5	19001

POWERSHELL SCRIPT - DEACTIVATE	A PowerShell script has been deactivated	5	19001
POWERSHELL SCRIPT - EDIT	A PowerShell script has been edited	8	19001
POWERSHELL SCRIPT - REACTIVATE	A PowerShell script has been reactivated	6	19001
POWERSHELL SCRIPT - VIEW	A PowerShell script has been viewed	5	19001
ROLE - ASSIGN USER OR GROUP	A role has been assigned to a user or group	5	19001
ROLE - CREATE	A role has been created	5	19001
ROLE - UNASSIGN USER OR GROUP	A role has been unassigned to a user or group	5	19001
ROLE PERMISSION - ADDED TO ROLE	A permission has been added to a role	5	19001
ROLE PERMISSION - REMOVED FROM ROLE	A permission has been removed from a role	5	19001
SECRET - ACCESS APPROVED	Access to a Secret has been approved	2	19001
SECRET - ACCESS DENIED	Access to a Secret has been denied	6	19001
SECRET - CHECKIN	A Secret has been checked in	1	19001
SECRET - CHECKOUT	A Secret has been checked out	5	19001
SECRET - COPY	A Secret has been copied	1	19001
SECRET - CREATE	A Secret has been created	1	19001
SECRET - CUSTOM AUDIT	A custom audit has been created	1	19001
SECRET - CUSTOM REQUIREMENT ADDED	A custom password requirement has been added to a Secret	2	19001
SECRET - CUSTOM REQUIREMENT REMOVED	A custom password requirement has been removed from a Secret	6	19001
SECRET - DELETE	A Secret has been deleted	5	19001
SECRET - DEPENDENCY ADDED	A dependency has been added	8	19001
SECRET - DEPENDENCY FAILURE	A dependency is missing	5	19001
SECRET - DEPENDENCY REMOVED	A dependency has been removed	8	19001
SECRET - EDIT	A Secret has been edited	5	19001
SECRET - EDIT VIEW	A Secrets view option has been edited	8	19001
SECRET - EXPIRES 1 DAY	A Secret expires in 1 day	5	19001
SECRET - EXPIRES 15 DAYS	A Secret expires in 15 days	1	19001
SECRET - EXPIRES 3 DAYS	A Secret expires in 3 days	1	19001
SECRET - EXPIRES 7 DAYS	A Secret expires in 7 days	1	19001
SECRET - EXPIRES TODAY	A Secret expires today	1	19001
SECRET - HEARTBEAT FAILURE	Heartbeat has not been detected for over 10 seconds	5	19001
SECRET - HEARTBEATSUCCESS	Heartbeat has been detected	1	19001
SECRET - HOOK CREATE	A hook has been created	3	19001
SECRET - HOOK DELETE	A hook has been deleted	8	19001
SECRET - HOOK EDIT	A hook has been edited	6	19001
SECRET - HOOKFAILURE	A hook has failed to initialize a PowerShell script	8	19001

SECRET - HOOKSUCCESS	A hook has successfully initialized a PowerShell script	1	19001
SECRET - LAUNCH	A Secret has been launched	1	19001
SECRET - PASSWORD COPIED TO CLIPBOARD	A password has been copied to the clipboard	5	19001
SECRET - PASSWORD_DISPLAYED	A Secret password has been displayed	5	19001
SECRET - SECRET POLICY CHANGE	The Secret policy assigned to a Secret has been changed	8	19001
SECRET - SESSION RECORDING VIEW	A Secret recording is being viewed	5	19001
SECRET - UNDELETE	A Secret has been restored	1	19001
SECRET - VIEW	A Secret has been viewed	1	19001
SECRET POLICY - CREATE	A Secret policy has been created	1	19001
SECRET POLICY - EDIT	A Secret policy has been edited	6	19001
SECRET TEMPLATE - COPY	A Secret template has been copied	1	19001
SECRET TEMPLATE - CREATE	A Secret template has been created	1	19001
SECRET TEMPLATE - EDIT	A Secret template has been edited	1	19001
SECRET TEMPLATE - FIELD ENCRYPTED	A field in a template has been encrypted	1	19001
SECRET TEMPLATE - FIELD EXPOSED	A field in a template has been exposed	6	19001
SECRETS EXPORTD	Secrets have been exported	10	19001
SECRETS IMPORTED	Secrets have been imported	1	19001
SYSTEM LOG	Thycotic Secret server system logs	1	19001
UNLIMITED ADMIN - DISABLED	Unlimited admin has been disabled	10	19001
UNLIMITED ADMIN - ENABLED	Unlimited admin has been enabled	10	19001
USER - ADDED TO GROUP	A user account has been added to a group	8	19001
USER - CREATE	A user account has been created		
USER - DISABLE	A user account has been disabled	5	19001
USER - ENABLE	A user account has been enabled	5	19001
USER - LOCKOUT	A user account has been locked out see payload for information	10	19001
USER - LOGIN	A user has logged on	1	19001
USER - LOGIN FAILURE	A user has entered an incorrect password	8	19001
USER - LOGOUT	A user has logged out	1	19001
USER - PASSWORD CHANGE	A users password has been changed	8	19001
USER - REMOVED FROM GROUP	A user account has been removed from a group	5	19001
USERAUDIT - EXPIRENOW	All Secrets a user has accessed have expired	5	19001