

Implementation Tech Note:

Integrating CyberArk Vault and Privileged Session Manager (PSM) with jSonar's SonarG/C2

jSonar's SonarG and SonarC2 systems can consume data from CyberArk systems in order to link fire id and ticketing details with audit data collected from various sources, including Database Activity Monitoring (DAM) systems, database native audit logs, operating system logs and others. This TechNote will outline the steps necessary to integrate the two systems and provide examples of typical use cases.

There are two primary components to this CyberArk integration note:

1. Receiving the data from CyberArk (system integration)
2. Using the data from CyberArk to enrich audit context (data integration)

System Integration

Integration between the systems is done using SonarSyslog, one of the SonarG/C2 facilities. CyberArk is configured to generate syslog data using a CEF format that is then sent to SonarG/C2. This data is automatically parsed and transformed into JSON documents that live in the cyberark_psm collection within the sonargd database.

Steps to integrate the systems:

1. Put jsonar.xml in your CyberArk vault syslog directory (see Appendix A)
2. Edit the dbparm.ini in your CyberArk vault to point to the SonarSyslog IP address and port, to monitor the appropriate CyberArk codes and to select the xml template file:

```
[SYSLOG]
UseLegacySyslogFormat=Yes
SyslogTranslatorFile=Syslog\jsonar.xml
SyslogServerPort=10540
SyslogServerProtocol=TCP
SyslogServerIP=34.196.101.5
SyslogMessageCodeFilter=308,302,300,295,319,320,378,380
```

3. Put cyberark.json in /etc/sonar/sonarsyslog on the jSonar machine and edit the appropriate port and connectivity definitions as needed:

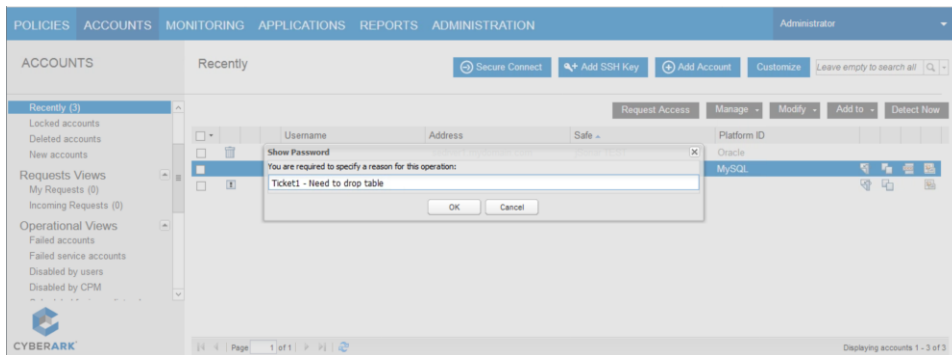
```
{
  "global_settings": {
    "do_not_ingest": [
      "cyberark_psm"
    ]
  }
}
```

```

    }
  },
  {
    "output_connection": {
      "event_format": {
        "standard": "CEF"
      },
      "sonar_URI": "<SonarW URI>",
      "target_db": "sonargd",
      "default_collection": "cyberark_psm",
      "unique_label": "cyberark_cef",
      "group_label": "cyberark_cef",
      "collection_selectors": {
        "select_by_key": [
          "act"
        ]
      },
      "redact_unmatched_fields": false
    },
    "timestamp": {
      "type": 9
    }
  }
}

```

At this point all CyberArk events , filtered by the appropriate MessageCodes, will be sent and processed by SonarG/C2. For example, consider a user retrieving a password via the CyberArk process shown below:



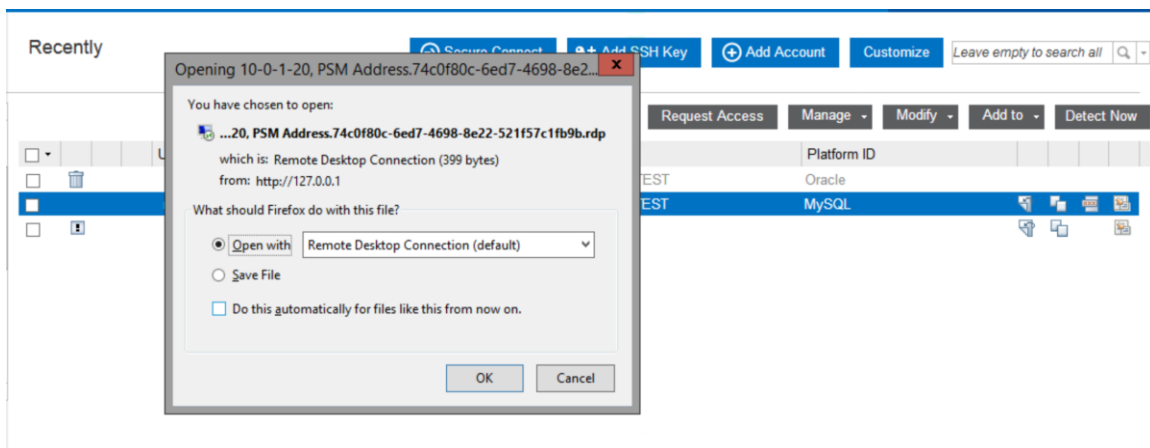
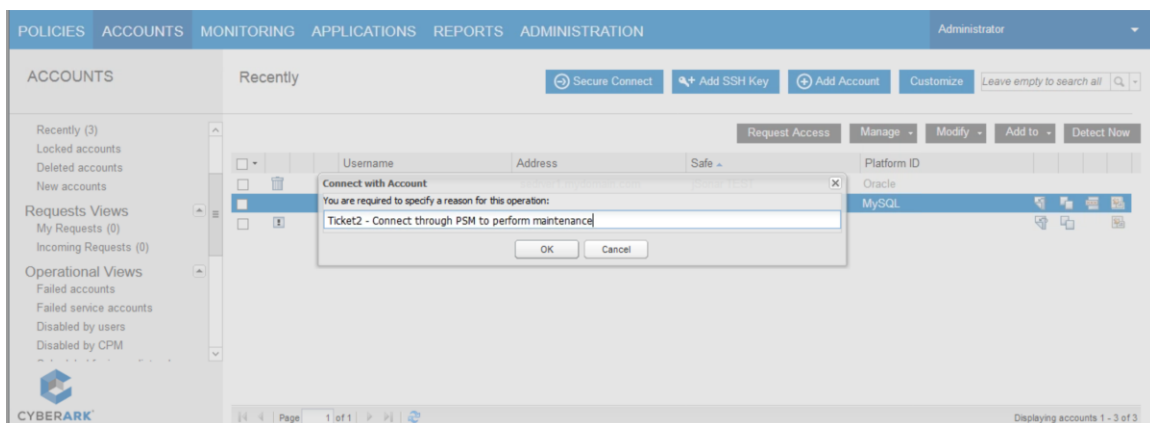
This action will generate a JSON record that records the IP of the target system (10.0.1.20 in this example), the privileged account (root in this example), the user retrieving the password (Administrator in this example) and the message entered within CyberArk (Ticket1 – Need to drop table).

```

{
  "_id" : ObjectId("5991993bbae694c251626a92"),
  "CEF Version" : 0,
  "Device Product" : "Vault",
  "Device Vendor" : "Cyber-Ark",
  "Device Version" : "9.90.0000",
  "Name" : "Retrieve password",
  "Severity" : "5",
  "Signature ID" : "295",
  "SonarG Source" : "VAULT",
  "deviceAction" : "Retrieve password",
  "applicationProtocol" : null,
  "Request Id" : null,
  "Ticket Id" : null,
  "Affected User Name" : null,
  "Safe Name" : "jSonar TEST",
  "Device Type" : "Database",
  "Database" : null,
  "Other info" : "10.0.1.20",
  "destinationHostName" : "10.0.1.20",
  "destinationUserName" : "root",
  "deviceAddress" : "10.0.1.20",
  "externalId" : null,
  "fileName" : "Root\\Database-MySQL-10.0.1.20-root",
  "message" : "(Action: Show Password)Ticket1 - Need to drop table#015",
  "reason" : null,
  "sourceHostName" : "127.0.0.1",
  "sourceUserName" : "Administrator",
  "timestamp" : ISODate("2017-08-14T12:36:06Z")
}

```

Similarly, if PSM is used for a pass-through connection as follows:



```

C:\Program Files\MySQL\MySQL Shell 1.0\bin\mysqlsh
mysqlx: [Warning] Using a password on the command line interface can be insecure
Creating a Session to 'root@10.0.1.20'
Classic Session successfully established. No default schema selected.
Welcome to MySQL Shell 1.0.9

Copyright (c) 2016, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type '\help', '\h' or '\?' for help, type '\quit' or '\q' to exit.

Currently in JavaScript mode. Use \sql to switch to SQL mode and execute queries.
mysql-js> _

```

the appropriate JSON record is generated and passed to SonarG/C2 as shown below:

```

{
  "_id" : ObjectId("59919b32bae694e5f84b273b"),
  "CEF Version" : 0,
  "Device Product" : "Vault",
  "Device Vendor" : "Cyber-Ark",
  "Device Version" : "9.90.0000",
  "Name" : "Use Password",
  "Severity" : "5",
  "Signature ID" : "308",
  "SonarG Source" : "VAULT",
  "deviceAction" : "Use Password",
  "applicationProtocol" : null,
  "Request Id" : null,
  "Ticket Id" : null,
  "Affected User Name" : null,
  "Safe Name" : "jSonar TEST",
  "Device Type" : "Database",
  "Database" : null,
  "Other info" : "10.0.1.20",
  "destinationHostName" : "10.0.1.20",
  "destinationUserName" : "root",
  "deviceAddress" : "10.0.1.20",
  "externalId" : null,
  "fileName" : "Root\\Database-MySQL-10.0.1.20-root",
  "message" : "(Action: Connect)Ticket2 - Connect through PSM to perform maintenance #015",
  "reason" : null,
  "sourceHostName" : "127.0.0.1",
  "sourceUserName" : "Administrator",
  "timestamp" : ISODate("2017-08-14T12:44:27Z")
}

```

Additional options are detailed in Appendix B.

Data Enrichment

In order to provide richer context and filtering around the events, CyberArk information can easily be joined with SonarG/C2 audit data. This will usually be done using the server IP information along with a specific time window. For example, one customer may require all relevant CyberArk data for the 2 hours before the audit event while another customer may allow up to 3 hours. One customer may require the closest record matching the audit record while another customer may prefer to get all possible password checkouts. Data enrichment therefore uses a template that can be fine-tuned during the implementation.

The first step is to identify the report for privileged user activity that you want to enrich with CyberArk data. The query is then extended with a \$join operation in order to make use of this data. Note that you must use a \$outer join type since you cannot be sure that the CyberArk data is available for each operation. Moreover, a null value in the CyberArk fields caused by the outer join is also a good candidate for an alert.

Typical conditions requested during implementation:

- Use the CyberArk record closest by timestamp to the audit event
- Use the CyberArk record closest by timestamp to the audit event so long as it is less than the audit event (i.e. password checkout was performed prior to the event)
- Similar, but password checkout could not have been earlier than 2 hours from the audit event

In all cases the join will include the DB User Name and the Server IP. In all cases the filter will include a condition of the form deviceAction: \$in: ["Use Password", "Retrieve Password"] and the field projected will include sourceUserName and message. Note that when using PSM it is possible to use the "Use Password" event or the "PSM Password" event but since the message exists in the Use Password event and not in the PSM Connect event the former is preferred.

Example:

In this example there are multiple CyberArk events all for the same user/server, but with different times. Using a prebuilt filter, the desired event can be easily isolated.

```
{
  "_id" : ObjectId("59919dfefbae6941fd1198e32"),
  "CEF Version" : 0,
  "Device Product" : "Vault",
  "Device Vendor" : "Cyber-Ark",
  "Device Version" : "9.90.0000",
  "Name" : "Retrieve password",
  "Severity" : "5",
  "Signature ID" : "295",
  "SonarG Source" : "VAULT",
  "deviceAction" : "Retrieve password",
  "applicationProtocol" : null,
  "Request Id" : null,
  "Ticket Id" : null,
  "Affected User Name" : null,
  "Safe Name" : "jSonar TEST",
  "Device Type" : "Database",
  "Database" : null,
  "Other info" : "10.0.1.20",
  "destinationHostName" : "10.0.1.20",
  "destinationUserName" : "root",
  "deviceAddress" : "10.0.1.20",
```

```
    "externalId" : null,  
    "fileName" : "Root\\Database-MySQL-10.0.1.20-root",  
    "message" : "(Action: Show Password)111#015",  
    "reason" : null,  
    "sourceHostName" : "127.0.0.1",  
    "sourceUserName" : "Administrator",  
    "timestamp" : ISODate("2017-08-14T12:56:25Z")  
  }  
{
```

```
    "_id" : ObjectId("5991ae6fbae6941fd1198ebf"),  
    "CEF Version" : 0,  
    "Device Product" : "Vault",  
    "Device Vendor" : "Cyber-Ark",  
    "Device Version" : "9.90.0000",  
    "Name" : "Retrieve password",  
    "Severity" : "5",  
    "Signature ID" : "295",  
    "SonarG Source" : "VAULT",  
    "deviceAction" : "Retrieve password",  
    "applicationProtocol" : null,  
    "Request Id" : null,  
    "Ticket Id" : null,  
    "Affected User Name" : null,  
    "Safe Name" : "jSonar TEST",  
    "Device Type" : "Database",  
    "Database" : null,  
    "Other info" : "10.0.1.20",  
    "destinationHostName" : "10.0.1.20",  
    "destinationUserName" : "root",  
    "deviceAddress" : "10.0.1.20",  
    "externalId" : null,  
    "fileName" : "Root\\Database-MySQL-10.0.1.20-root",  
    "message" : "(Action: Show Password)Ticket5#015",  
    "reason" : null,  
    "sourceHostName" : "127.0.0.1",  
    "sourceUserName" : "Administrator",  
    "timestamp" : ISODate("2017-08-14T14:06:34Z")  
  }  
{
```

```
    "_id" : ObjectId("5991aea2bae6941fd1198ec9"),  
    "CEF Version" : 0,  
    "Device Product" : "Vault",  
    "Device Vendor" : "Cyber-Ark",  
    "Device Version" : "9.90.0000",  
    "Name" : "Use Password",  
    "Severity" : "5",  
    "Signature ID" : "308",  
    "SonarG Source" : "VAULT",  
    "deviceAction" : "Use Password",  
    "applicationProtocol" : null,  
    "Request Id" : null,  
    "Ticket Id" : null,  
    "Affected User Name" : null,  
    "Safe Name" : "jSonar TEST",  
    "Device Type" : "Database",  
    "Database" : null,  
    "Other info" : "10.0.1.20",  
    "destinationHostName" : "10.0.1.20",  
    "destinationUserName" : "root",  
    "deviceAddress" : "10.0.1.20",  
    "externalId" : null,  
    "fileName" : "Root\\Database-MySQL-10.0.1.20-root",  
    "message" : "(Action: Connect)Ticket6 #015",  
    "reason" : null,  
    "sourceHostName" : "127.0.0.1",  
    "sourceUserName" : "Administrator",
```

```

    "timestamp" : ISODate("2017-08-14T14:07:23Z")
  }
  {
    "_id" : ObjectId("5991aea2bae6941fd1198eca"),
    "CEF Version" : 0,
    "Device Product" : "Vault",
    "Device Vendor" : "Cyber-Ark",
    "Device Version" : "9.90.0000",
    "Name" : "PSM Connect",
    "Severity" : "5",
    "Signature ID" : "300",
    "SonarG Source" : "VAULT",
    "deviceAction" : "PSM Connect",
    "applicationProtocol" : "SQLNet",
    "Request Id" : null,
    "Ticket Id" : null,
    "Affected User Name" : null,
    "Safe Name" : "jSonar TEST",
    "Device Type" : "Database",
    "Database" : null,
    "Other info" : "      ",
    "destinationHostName" : "10.0.1.20",
    "destinationUserName" : "root",
    "deviceAddress" : null,
    "externalId" : "2cb49414-866d-4241-8521-b33b1b9376fe",
    "fileName" : "Root\\Database-MySQL-10.0.1.20-root",
    "message" : "#015",
    "reason" : null,
    "sourceHostName" : "127.0.0.1",
    "sourceUserName" : "Administrator",
    "timestamp" : ISODate("2017-08-14T14:07:25Z")
  }
}

```

Assume that we have a database audit record from 2017-08-14 14:06:34 showing privileged activity on Server IP 10.0.1.20 via DB Username “root.” To get the CyberArk record closest in time to this audit event, but prior to the actual event (i.e. password checkout was performed prior to the event) and that no longer than 2 hours have passed since the event, use the following pipeline:

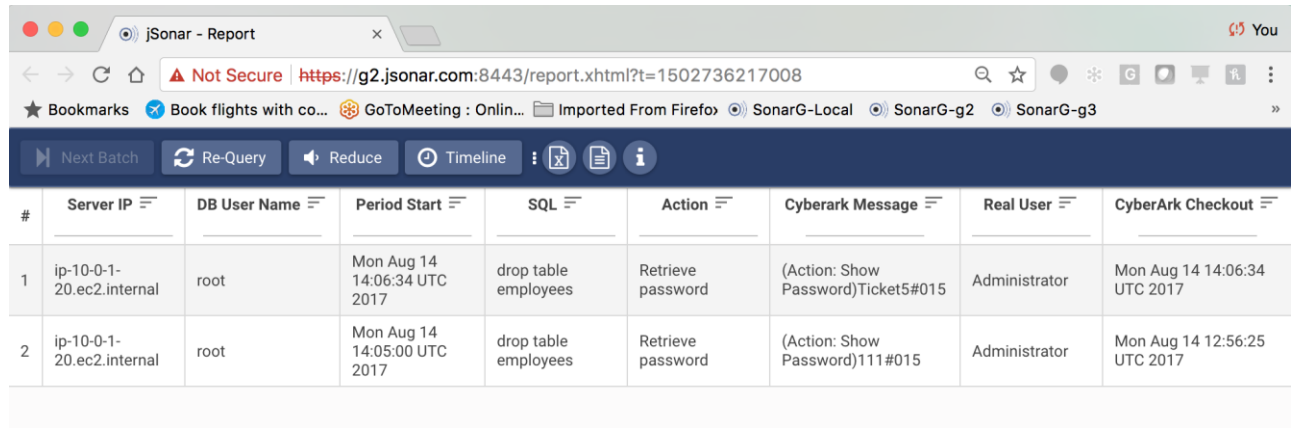
```

{$join: {
  $joined: {cyberark: "cyberark_psm"},
  $outer: ["cyberark"],
  $selector: {cyberark: {deviceAction: {$in: ["Use Password", "Retrieve Password"]}}},
  $match: [
    {"DB User Name": "$cyberark.$destinationUserName"},
    {"Server IP": "$cyberark.$destinationHostName"},
    { $value: "$Period Start",
      $gte: "$cyberark.$timestamp",
      $lte: {$add: ["$cyberark.$timestamp", 7200000 ]}}
  ], $project: {timeDiff: {$subtract: ["$Period Start", "$cyberark.$timestamp"]}},
  $group : {mostRecentCheckout : { $minDoc : "$timeDiff"}}
}}

```

Using the three CyberArk records listed above and multiple audit events, including an audit event at 14:15 and another audit event at 14:06 we can automatically

reconcile the CyberArk events with the audit events that were captured from the DB monitoring system, thereby eliminating a lot of manual effort while improving security and compliance processes.



#	Server IP	DB User Name	Period Start	SQL	Action	Cyberark Message	Real User	CyberArk Checkout
1	ip-10-0-1-20.ec2.internal	root	Mon Aug 14 14:06:34 UTC 2017	drop table employees	Retrieve password	(Action: Show Password)Ticket5#015	Administrator	Mon Aug 14 14:06:34 UTC 2017
2	ip-10-0-1-20.ec2.internal	root	Mon Aug 14 14:05:00 UTC 2017	drop table employees	Retrieve password	(Action: Show Password)111#015	Administrator	Mon Aug 14 12:56:25 UTC 2017

Appendix A – jsonar.xsl - Template File for Generating CEF Syslog Events

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:import href='../Syslog/RFC5424Changes.xsl'/>
  <xsl:output method='text' version='1.0' encoding='UTF-8'/>

  <xsl:template match="/">
    <xsl:apply-imports />
    <xsl:for-each select="syslog/audit_record">: CEF:0|<xsl:value-of
select="Vendor"/>|<xsl:value-of select="Product"/>|<xsl:value-of
select="Version"/>|<xsl:value-of select="MessageID"/>|<xsl:choose><xsl:when
test="Severity='Critical' or Severity='Error'">Failure:
</xsl:when></xsl:choose><xsl:call-template name="string-replace">
  <xsl:with-param name="from" select=""/>
  <xsl:with-param name="to" select="\\"/>
  <xsl:with-param name="string" select="Desc"/>
</xsl:call-template>|<xsl:choose><xsl:when
test="Severity='Critical'">10</xsl:when><xsl:when
test="Severity='Error'">7</xsl:when><xsl:when
test="Severity='Info'">5</xsl:when><xsl:otherwise>0</xsl:otherwise></xsl:choose
><!--xsl:value-of select="Severity"/-->|act=<xsl:call-template name="string-
replace">
  <xsl:with-param name="from" select=""/>
  <xsl:with-param name="to" select="\\"/>
  <xsl:with-param name="string" select="Action"/>
</xsl:call-template> suser=<xsl:call-template name="string-replace">
  <xsl:with-param name="from" select=""/>
  <xsl:with-param name="to" select="\\"/>
  <xsl:with-param name="string" select="Issuer"/>
</xsl:call-template> fname=<xsl:call-template name="string-replace">
  <xsl:with-param name="from" select=""/>
  <xsl:with-param name="to" select="\\"/>
  <xsl:with-param name="string" select="File"/>
</xsl:call-template> dvc=<xsl:call-template name="string-replace">
  <xsl:with-param name="from" select=""/>
  <xsl:with-param name="to" select="\\"/>
  <xsl:with-param name="string" select="GatewayStation"/>
</xsl:call-template> shost=<xsl:choose>
  <!--xsl:If its PSM Connect and Disconnect event we will show SrcHost value
otherwise we will show station value"/-->
  <xsl:when test="MessageID=300 or MessageID=301 or MessageID=302 or
MessageID=303"><xsl:call-template name="string-GetValue">
  <xsl:with-param name="from" select="SrcHost"/>
  <xsl:with-param name="to" select="';'"/>
```

```

        <xsl:with-param name="string" select="ExtraDetails"/>
    </xsl:call-template></xsl:when>
    <xsl:otherwise><xsl:call-template name="string-replace">
        <xsl:with-param name="from" select=""/>
        <xsl:with-param name="to" select="\\"/>
        <xsl:with-param name="string" select="Station"/>
    </xsl:call-template></xsl:otherwise>
</xsl:choose> dhost=<xsl:choose>
    <!--xsl:If its PSM Connect and Disconnect event we will show DstHost value/-->
>
    <xsl:when test="MessageID=300 or MessageID=301 or MessageID=302 or
MessageID=303"><xsl:call-template name="string-GetValue">
    <xsl:with-param name="from" select="DstHost="/>
    <xsl:with-param name="to" select="';"/>
    <xsl:with-param name="string" select="ExtraDetails"/>
</xsl:call-template></xsl:when>
    <!--xsl:For transparent connection event we will show RemoteMachine value
from the PVWA XML/-->
    <xsl:when test="MessageID=295 and
PvwaDetails/RequestReason/ConnectionDetails/RemoteMachine!=""><xsl:call-
template name="string-replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select="\\"/>
    <xsl:with-param name="string"
select="PvwaDetails/RequestReason/ConnectionDetails/RemoteMachine"/>
</xsl:call-template></xsl:when>
    <!--xsl:Check if extra details is not empty is so extract the dsthost value from
it/-->
    <xsl:when test="ExtraDetails!=""><xsl:call-template name="string-
GetValue">
    <xsl:with-param name="from" select="DstHost="/>
    <xsl:with-param name="to" select="';"/>
    <xsl:with-param name="string" select="ExtraDetails"/>
</xsl:call-template></xsl:when>
    <!--xsl:Otherwise we will show Address value from the file categories/-->
    <xsl:otherwise><xsl:for-each select="CAProperties/CAProperty"><xsl:if
test="@Name='Address'"><xsl:call-template name="string-replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select=""/>
    <xsl:with-param name="string" select="@Value"/></xsl:call-
template></xsl:if></xsl:for-each>
    </xsl:otherwise>
</xsl:choose> duser=<xsl:choose>
    <!--xsl:If its PSM Connect and Disconnect event we will show User value/-->
    <xsl:when test="MessageID=300 or MessageID=301 or MessageID=302 or
MessageID=303"><xsl:call-template name="string-GetValue">

```

```

        <xsl:with-param name="from" select="'User='"/>
        <xsl:with-param name="to" select="';'"/>
        <xsl:with-param name="string" select="ExtraDetails"/>
        <!--xsl:If it not PSM we check if the Target user field is not empty if so we
show his value/-->
        </xsl:call-template></xsl:when><xsl:when test="TargetUser != ''">
            <xsl:call-template name="string-replace">
                <xsl:with-param name="from" select="'='"/>
                <xsl:with-param name="to" select="'\'"/>
                <xsl:with-param name="string" select="TargetUser"/></xsl:call-
template></xsl:when>
        <!--xsl:Otherwise we show the username value from the file categories/-->
        <xsl:otherwise><xsl:for-each select="CAProperties/CAProperty"><xsl:if
test="@Name='UserName'"><xsl:call-template name="string-replace">
            <xsl:with-param name="from" select="'='"/>
            <xsl:with-param name="to" select="'/='"/>
            <xsl:with-param name="string" select="@Value"/></xsl:call-
template></xsl:if></xsl:for-each>
        </xsl:otherwise></xsl:choose> externalId=<xsl:call-template name="string-
GetValue">
            <xsl:with-param name="from" select="'SessionID='"/>
            <xsl:with-param name="to" select="';'"/>
            <xsl:with-param name="string" select="ExtraDetails"/>
        </xsl:call-template> app=<xsl:call-template name="string-GetValue">
            <xsl:with-param name="from" select="'Protocol='"/>
            <xsl:with-param name="to" select="';'"/>
            <xsl:with-param name="string" select="ExtraDetails"/>
        </xsl:call-template> reason=<xsl:call-template name="string-GetValue">
            <xsl:with-param name="from" select="'Command='"/>
            <xsl:with-param name="to" select="';'"/>
            <xsl:with-param name="string" select="ExtraDetails"/>
        </xsl:call-template> cs1Label=Affected User Name cs1=<xsl:call-template
name="string-replace">
            <xsl:with-param name="from" select="'='"/>
            <xsl:with-param name="to" select="'\='"/>
            <xsl:with-param name="string" select="SourceUser"/>
        </xsl:call-template> cs2Label=Safe Name cs2=<xsl:call-template name="string-
replace">
            <xsl:with-param name="from" select="'='"/>
            <xsl:with-param name="to" select="'\='"/>
            <xsl:with-param name="string" select="Safe"/>
        </xsl:call-template> cs3Label=Device Type cs3=<xsl:for-each
select="CAProperties/CAProperty"><xsl:if test="@Name='DeviceType'"><xsl:call-
template name="string-replace">
            <xsl:with-param name="from" select="'='"/>
            <xsl:with-param name="to" select="'/='"/>

```

```

        <xsl:with-param name="string" select="@Value"/>
    </xsl:call-template></xsl:if></xsl:for-each> cs4Label=Database cs4=<xsl:call-
template name="string-GetValue">
    <xsl:with-param name="from" select="DataBase=""/>
    <xsl:with-param name="to" select="";"/>
    <xsl:with-param name="string" select="ExtraDetails"/>
</xsl:call-template> cs5Label=Other info cs5=<xsl:call-template name="string-
replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select="\=""/>
    <xsl:with-param name="string" select="Location"/>
</xsl:call-template> <xsl:call-template name="string-replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select="\=""/>
    <xsl:with-param name="string" select="Category"/>
</xsl:call-template> <xsl:call-template name="string-replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select="\=""/>
    <xsl:with-param name="string" select="GatewayStation"/>
</xsl:call-template>
    cs6Label=Request Id cs6=<xsl:value-of select="RequestId"/>
    cs7Label=Ticket Id cs7=<xsl:value-of select="Reason"/>
<!--
    rt=<xsl:value-of select="Timestamp"/>
-->
    msg=<xsl:call-template name="string-replace">
    <xsl:with-param name="from" select=""/>
    <xsl:with-param name="to" select="\=""/>
    <xsl:with-param name="string" select="Reason"/>
    </xsl:call-template> <xsl:choose><xsl:when test="Severity='Critical' or
Severity='Error'">Failure: </xsl:when></xsl:choose>
    </xsl:for-each>
<xsl:text>&#10;</xsl:text>
</xsl:template>

```

<!-- Gets the Value of a member from a long string
 from - the name of the member(pre)
 to - this represents the end of the value(post)
 its also calls the string-replace and replace the = with /=
 Parsing needed for Arcsight.-->

```

<xsl:template name="string-GetValue" >
    <xsl:param name="string"/>
    <xsl:param name="from"/>
    <xsl:param name="to"/>
    <xsl:choose>
        <xsl:when test="contains($string,$from)">

```

```

        <xsl:call-template name="string-replace">
            <xsl:with-param name="string" select="substring-before(substring-
after($string,$from),$to)"/>
            <xsl:with-param name="from" select=""/>
            <xsl:with-param name="to" select=""/>
        </xsl:call-template>
    </xsl:when>
</xsl:choose>
</xsl:template>

<!-- replace all occurrences of the character(s) `from'
by the string `to' in the string `string'.-->
<xsl:template name="string-replace" >
    <xsl:param name="string"/>
    <xsl:param name="from"/>
    <xsl:param name="to"/>
    <xsl:choose>
        <xsl:when test="contains($string,$from)">
            <xsl:value-of select="substring-before($string,$from)"/>
            <xsl:value-of select="$to"/>
            <xsl:call-template name="string-replace">
                <xsl:with-param name="string" select="substring-after($string,$from)"/>
                <xsl:with-param name="from" select="$from"/>
                <xsl:with-param name="to" select="$to"/>
            </xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="$string"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

</xsl:stylesheet>

```

Appendix B – Additional Options for System Integration

1. Eliminating events from built-in CyberArk users:

Some of the events sent by CyberArk pertain to internal users. For example, the following record has a PVWAppUser:

```

Aug 15 10:09:13 VAULT : CEF:0|Cyber-Ark|Vault|9.90.0000|295|Retrieve
password|5|act=Retrieve password suser=PVWAppUser fname=Root\servicenow-user dvc=
shost=10.0.1.20 dhost=NA duser=admin externalId= app= reason= cs1Label=Affected User
Name cs1= cs2Label=Safe Name cs2=ServiceNow cs3Label=Device Type cs3=Operating

```

System cs4Label=Database cs4= cs5Label=Other info cs5= cs6Label=Request Id cs6=
cs7Label=Ticket Id cs7=Validate ticket number [SIR0001000], User [Administrator], Account
[ServiceNow-servicenow-user] msg=Validate ticket number [SIR0001000], User
[Administrator], Account [ServiceNow-servicenow-user]#015

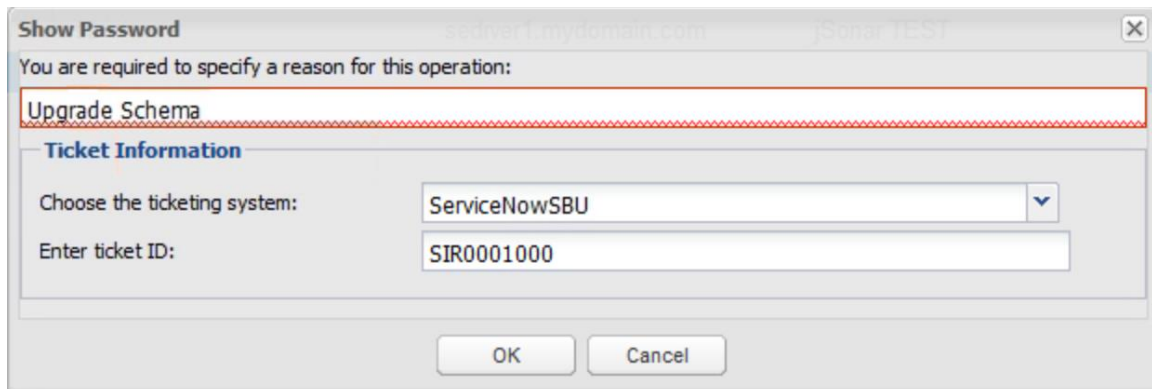
There is no issue with ingesting these records into SonarG/C. If you wish to filter them out add the following to the JSON specification to force these records to be thrown away:

```
{
  "output_connection": {
    "event_format": {
      "standard": "CEF"
    },
    "sonar_URI": "mongodb://admin:sonarw321@127.0.0.1:27117/admin",
    "target_db": "sonargd",
    "default_collection": "$null",
    "unique_label": "cyberark_cef_drop_internal_users",
    "group_label": "cyberark_cef",
    "collection_selectors": {
      "select_by_key_value": {
        "suser": "PVWAApUser"
      }
    },
    "redact_unmatched_fields": false
  },
}
```

Note that you can use a more complex condition to throw out records with multiple suser values. Note also that in your CyberArk environment the user names used internally may be different.

2. Extracting Ticket IDs:

CyberArk has the ability to validate a ticket number before allowing a user access to a database account and that ticket number is a part of the syslog record. For example, CyberArk can be configured to require a ServiceNow activity when retrieving a password as follows:



In this case the syslog message incoming to SonarSyslog looks like:

```
Aug 15 10:53:16 VAULT : CEF:0|Cyber-Ark|Vault|9.90.0000|295|Retrieve
password|5|act=Retrieve password suser=Administrator fname=Root\Database-
MySQL-10.0.1.20-root dvc=10.0.1.20 shost=127.0.0.1 dhost=10.0.1.20 duser=root
externalId= app= reason= cs1Label=Affected User Name cs1= cs2Label=Safe Name
cs2=jSonar TEST cs3Label=Device Type cs3=Database cs4Label=Database cs4=
cs5Label=Other info cs5=10.0.1.20 cs6Label=Request Id cs6=
cs7Label=Ticket Id cs7=(Action: Show Password)(Ticket ID=SIR0001000)(Ticketing
System=ServiceNowSBU)(Ticketing Audit=sn_si_incident) Upgrade Schema
msg=(Action: Show Password)(Ticket ID=\SIR0001000)(Ticketing
System\=ServiceNowSBU)(Ticketing Audit\=sn_si_incident) Upgrade Schema#015
```

You can then use SonarSyslog's parsing capabilities to extract the ticket number to a separate field by adding this to your JSON configuration:

```
"Ticket Number": {
  "generate": true,
  "eval": [
    "regex",
    "cs7",
    ".*Ticket ID=([a-zA-Z0-9]*).*"
  ]
}
```

The resulting document looks like:

```
{
  "_id" : ObjectId("5995de34bae694f1bc6566c4"),
  "CEF Version" : 0,
  "Device Product" : "Vault",
  "Device Vendor" : "Cyber-Ark",
  "Device Version" : "9.90.0000",
  "Name" : "Retrieve password",
  "Severity" : "5",
  "Signature ID" : "295",
  "SonarG Source" : "VAULT",
  "deviceAction" : "Retrieve password",
  "applicationProtocol" : null,
  "Affected User Name" : null,
  "Safe Name" : "jSonar TEST",
  "Device Type" : "Database",
  "Database" : null,
  "Other info" : "10.0.1.20",
  "Request Id" : null,
  "Ticket Id" : "(Action: Show Password)(Ticket ID=SIR0001000)(Ticketing System=ServiceNowSBU)(Ticketing
Audit=sn_si_incident) Upgrade Schema",
  "destinationHostName" : "10.0.1.20",
  "destinationUserName" : "root",
  "deviceAddress" : "10.0.1.20",
  "fileName" : "Root\\Database-MySQL-10.0.1.20-root",
  "message" : "(Action: Show Password)(Ticket ID=SIR0001000)(Ticketing System=ServiceNowSBU)(Ticketing
Audit=sn_si_incident) Upgrade Schema",
  "sourceHostName" : "127.0.0.1",
  "sourceUserName" : "Administrator",
```

```
"timestamp" : ISODate("2017-08-17T18:19:27Z"),  
"Ticket Number" : "SIR0001000"
```

```
}
```