Useful Queries for the VMware Carbon Black Cloud Splunk App

VMware Empowering the Modern SOC

Table of contents

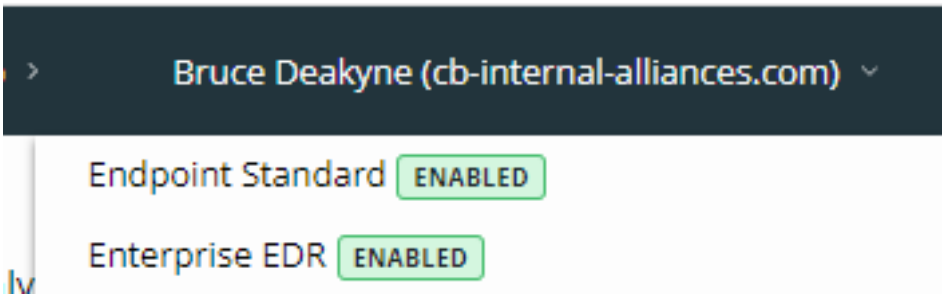
Useful Queries for the VMware Carbon Black Cloud Splunk App

Introduction

The VMware Carbon Black Cloud App brings visibility from VMware's endpoint protection capabilities into Splunk for visualization, reporting, detection, and threat hunting use cases. With so much data, your SOC can find endless opportunities for value. But sometimes, it's helpful to have a few examples to get started.

All queries use Splunk eventtypes and sourcetypes defined in the Carbon Black Cloud Splunk App. You can find more information on installing & configuring the app on the Developer Network Splunk Integration page. Each use case lists additional requirements, including which Carbon Black Cloud products your org must have enabled and which data sources, alert actions, and custom commands you need to configure in Splunk.

To determine which products your organization has enabled, in the Carbon Black Cloud console click your username in the upper-right corner. Enabled products will have the "ENABLED" tag.



Alert Triage

Investigating CB Analytics Alerts

Required Product: Endpoint Standard

Every CB Analytics (NGAV) Alert is tied to one or more Enriched Events. It's very helpful to pivot to these events (also known as "Alerted Events") and identify the behavior that led to the alert. This can provide insights like what processes, cmdlines, and users were involved if any network connections occurred, and what files or registry keys were modified.

The syntax varies depending on whether you're using the App Inputs & Alert Actions or Data Forwarder.

- Using App Input & Alert Actions
  - **Required Data/Configurations:** Alerts (App Input), Enrich CB Analytic Events (App Alert Action), CB Analytics - Ingest Enriched Events (Splunk Alert)
  - Follow the setup instructions from the App Configuration Video
    - Alerts Inputs (timestamp 5:17)
    - Alert Actions (timestamps 6:38 and 8:30).

- Using the Data Forwarder
  - **Required Data:** Alerts (Data Forwarder), Endpoint Events (Data Forwarder)
  - The Data Forwarder can be configured to push both Alerts and Endpoint Events into Splunk via an AWS S3 bucket. You can filter to send only Alerted Events, which are just a small fraction of all endpoint events, with the following Custom Query Filter:

```
alert_id:*
```

Using App Input & Alert Actions

```
eventtype="vmware_cbc_cb_analytics"
| eval alert_id = id
| dedup alert_id
| join alert_id [
  search eventtype="vmware_cbc_base_index" sourcetype="vmware:alert_action:vmware-enrich-events"
  | rename alert_id{} as alert_id
  | stats dc(event_id) as event_count, values(event_type) as event_types, values(process_cmdline{}) as
process_cmdlines, values(event_description) as enriched_event_descriptions by alert_id
]
| table alert_id, severity, device_name, reason, event_count, event_types, process_cmdlines,
enriched_event_descriptions
```

Using the Data Forwarder

```
eventtype="vmware_cbc_cb_analytics"
| eval alert_id = id
| dedup alert_id
| join alert_id [
  search eventtype="vmware_cbc_events" alert_id = *
  | stats
    dc(event_id) as event_count,
    values(type) as event_types,
    values(process_cmdline) as process_cmdlines,
    values(event_description) as enriched_event_descriptions
    by alert_id
]
| table alert_id, severity, device_name, reason, event_count, event_types, process_cmdlines,
enriched_event_descriptions
```

alert_id	severity	device_name	reason	event_count	event_types	process_cmdlines
eafdc064-fc4e-ea78-c5c2-db325b34e4d2	6	CBSTD-WKSH	The application powershell.exe was leveraged to make a potentially malicious network connection.	10	childproc crossproc filemod netconn	"C:\Program Files (x86)\Microsoft Office\Office15\EXCEL.EXE" "C:\Users\EndUser\Desktop\Credit_Cards_2018.xlsm" "C:\Wi ('JABFAGkAPQAnACQATQBoAD0AJwAnAFsARABsAGwASQBtAHAAbwByAHQAKAAoACIAbQBzAHYAYwByAHQALgAiACsAIgBkACIAKwAiAGwAbAAiACkAKQE "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -ec JABFAGkAPQAnACQATQBoAD0AJwAnAFsARABsAGwASQBtAHAAbwByAHQAKAAoACIAbQBzAHYAYwByAHQALgAiACsAIgBkACIAKwAiAGwAbAAiACkAKQBdA "C:\Windows\syswow64\Windowspowershell\v1.0\powershell.exe" -noexit -e JABNAGgAPQAnAFsARABsAGwASQBtAHAAbwByAHQAKAAoAC

alert_id	severity	device_name	reason	event_count	event_types	process_cmdlines	enriched_event_descriptions
8078c6d4-dcc8-105b-982f-88db7c50d8d8	5	QA\TESTSYS	The application svchost.exe invoked another application (taskhost.exe).	4	endpoint.event.apicall endpoint.event.procstart	"C:\Program Files\Tenable\Nessus Agent\nessus-service.exe" C:\Windows\System32\svchost.exe -k termsvcs C:\Windows\system32\svchost.exe -k netsvcs	The application "<share><link hash="6a468cc6dbfb0d904b692d55378 "CreateProcess". The operation was successful. The application \capture\RdpCoreTS-WppAutoTrace-<.etl", by calling the function "<share><link hash="b79e0890e5acffe7966bb32a6aaa415d6e334d0df54

Process Details for Watchlist Alerts

Required Product: Enterprise EDR

Required Data/Configurations: Alerts (App Input or Data Forwarder), Process GUID Details (App Alert Action), Process GUID Details (user-created Splunk Alert)

Carbon Black Cloud has dozens of metadata fields about every process that executes on an endpoint. While not all of it is included in a Watchlist Alert, the “Process GUID Details” Alert Action can automatically query Carbon Black Cloud for all process details following a Watchlist Alert. Follow the setup instructions from the App Configuration Video (timestamps 6:38 and 9:13). Any field tagged “DETAILS” in the Process Search Fields documentation will be available.

Watchlist Alerts with rich process metadata

```
eventtype="vmware_cbc_watchlist"
| join process_guid [
  search eventtype="vmware_cbc_action_index" sourcetype="vmware:alert_action:vmware-process-guid-details"
  | rename results{}.* as *
]
| table device_name, report_name, severity, process_cmdline{}, process_username{}, parent_cmdline
```

device_name	report_name	severity	process_cmdline{}	process_username{}	parent_cmdline
QA\TESTSYS	EXE_Source ips with severity 10	10	C:\WINDOWS\system32\svchost.exe -k NetworkService -p -s CryptSvc	NT AUTHORITY\NETWORK SERVICE	C:\WINDOWS\system32\services.exe
carbonblack-win1	Defense Evasion - Windows Event Command Line Utility Use	8	C:\WINDOWS\system32\cmd.exe /c "wevtutil gl "Security""	NT AUTHORITY\SYSTEM	C:\Program Files\test\test_conf_win_srv\test_conf_win_srv.exe

Handy URLs

Products Required: Any

Required Data: Alerts (App Input or Data Forwarder)

When investigating an alert, it can be helpful to pivot back to the Carbon Black Cloud console to view purpose-built NGAV and EDR content, such as the process tree. There are a variety of pages you may want to visit and those may vary team to team. These URLs can be formed based on the content of an Alert.

Each URL requires your Carbon Black Cloud console domain (e.g. `https://defense-prod05.conferdeploy.net`). If you’re using the App Input for alerts, the query will get it from the Splunk `host` field. If you’re using the Data Forwarder input for alerts, the query will get the console URL from the available `alert_url` field.

Carbon Black Cloud URLs for each alert

```
eventtype="vmware_cbc_alerts"
| rex field=alert_url "^(?<cbc_console>https:\/\/[^\/]+)\/.*"
| eval cbc_console = case(isnotnull(cbc_console), cbc_console, like(host, "%confer%.net%"), "https://" + host,
1=1, "https://replaceme")
| eval alert_page_url = cbc_console + "/alerts?selected[id]=" + id +
"&s[highlight]=true&s[c][query_string]=alert_id%3A" + id + "&orgKey=" + org_key
| eval alert_triage_url = if(type = "CB_ANALYTICS", cbc_console + "/triage?incidentId=" + id + "&orgKey=" +
org_key, "N/A")
| eval process_tree_url = if(type = "WATCHLIST", cbc_console + "/analyze?processGUID=" + process_guid +
"&alertId=" + id + "&deviceId=" + device_id + "&orgKey=" + org_key, "N/A")
| eval investigate_process_url = cbc_console + "/investigate/processes?query=alert_id%3A" + id + "&orgKey=" +
org_key
| eval investigate_events_url = cbc_console + "/investigate/events?query=alert_id%3A" + id + "&orgKey=" +
org_key
| eval endpoint_url = cbc_console + "/inventory/endpoints?s[query]=deviceId%3A" + device_id + "&orgKey=" +
org_key
| table *_url
```

alert_url	endpoint_url	investigate_events_url	investigate_p
https://defense.conferdeploy.net/cb/investigate/processes?orgId=1105&query=alert_id%3Add3a03a8-bec0-4545-bce1-abe67c6c1ce0+AND+device_id%3A4467271&searchWindow=ALL	https://defense.conferdeploy.net/inventory/endpoints?s[query]=deviceId%3A4467271&orgKey=ABCD1234	https://defense.conferdeploy.net/investigate/events?query=alert_id%3Add3a03a8-bec0-4545-bce1-abe67c6c1ce0&orgKey=ABCD1234	https://defe/processes?qorgKey=ABCD1
https://defense.conferdeploy.net/cb/investigate/processes?orgId=1105&query=alert_id%3Add14fccf-9f55-45da-9d84-53880c7e8997+AND+device_id%3A4467271&searchWindow=ALL	https://defense.conferdeploy.net/inventory/endpoints?s[query]=deviceId%3A4467271&orgKey=ABCD1234	https://defense.conferdeploy.net/investigate/events?query=alert_id%3Add14fccf-9f55-45da-9d84-53880c7e8997&orgKey=ABCD1234	https://defe/processes?qorgKey=ABCD1

More Readable Enriched Events

Products Required: Endpoint Standard

Required Data: Events (Data Forwarder) or Enrich CB Analytic Events (App Alert Action)

Enriched Event descriptions are preformatted for a user interface and contain some HTML markup. This can be effectively stripped away, though you may lose some context. For example:

```
The application "<share><link
hash="643ec58e82e0272c97c2a59f6020970d881af19c0ad5029db9c958c13b6558c7">C:\Windows\system32\svchost.exe -k netsvcs -p
-s Schedule</link></share>" invoked the application "<share><link
hash="f1f67830fc3531dfbdaf5315f59422438ab9f243d89491ac75d1818e7ed98b5d">C:\program files
(x86)\google\update\googleupdate.exe</link></share>". The operation was <accent>blocked by Cb Defense</accent>.
```

Becomes

```
The application "C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule" invoked the application "C:\program files
(x86)\google\update\googleupdate.exe". The operation was blocked by Cb Defense.
```

Remove tags from Enriched Events

```
eventtype="vmware_cbc_events" event_description="*"
| rex mode=sed field=event_description "s/(<[^>]+>)//g"
| table event_description
```

event_description
The application "C:\Windows\System32\svchost.exe -k NetworkService -p -s DoSvc" accepted a TCP/7680 connection from 10.176.186.147:53581 to 10.203.96.177:7680. The device was on the corporate network using the public address 65.111.99.2 (DEV01-37x-1.TEST.COM, located in Mountain View CA, United States). The operation was successful.
The application "C:\windows\syswow64\cmd.exe" invoked the application "C:\windows\system32\conhost.exe". The operation was successful.
The application "c:\windows\system32\conhost.exe" attempted to open the process "c:\windows\syswow64\cmd.exe", by calling the function "OpenProcess". The operation was successful.
The application "C:\Windows\system32\svchost.exe -k netsvcs" invoked the application "C:\windows\system32\taskeng.exe".

Alert Trends

Products Required: Any

Required Data: Alerts (App Input or Data Forwarder)

Aggregate data can help identify improvements in your SOC workflows. Are their known-good processes driving false positive alert volume? Can existing Watchlist Reports be tuned for greater efficacy? What type of alerts should new SOAR playbooks focus on?

Some of these visualizations are already built into the Carbon Black Cloud Splunk App's Alerts Overview dashboard, but these queries may enable your SOC to more easily customize what's important to you.

The data is available from both the data model and the indexed data. The data model queries should execute much faster, especially if you're looking for 7+ day trends in large environments. However, as they're less intuitive to edit, your team may choose to work from the indexed data queries for any customizations.

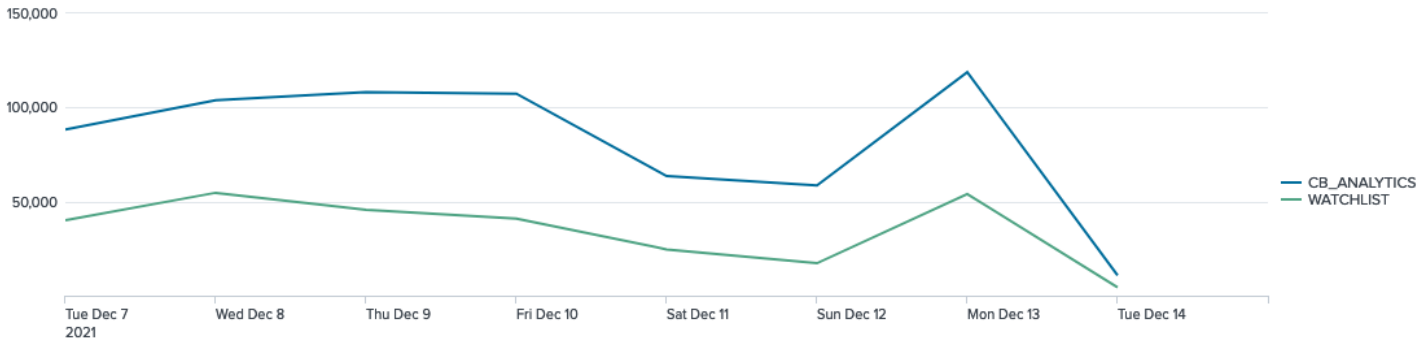
Alerts over time by type

Indexed Data

```
eventtype="vmware_cbc_alerts"
| timechart span=1d dc(id) by type
```

Data Model

```
| `vmware_tstats` earliest(All_CBC.Alerts.create_time) as create_time
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts
  by All_CBC.cbc_event_id, All_CBC.type
| rename All_CBC.* as *
| rename Alerts.* as *
| rename cbc_event_id as alert_id
| eval _time = strptime(create_time,"%Y-%m-%dT%H:%M:%S.%3QZ")
| timechart span=1d dc(alert_id) by type
```



Alerts over time by severity

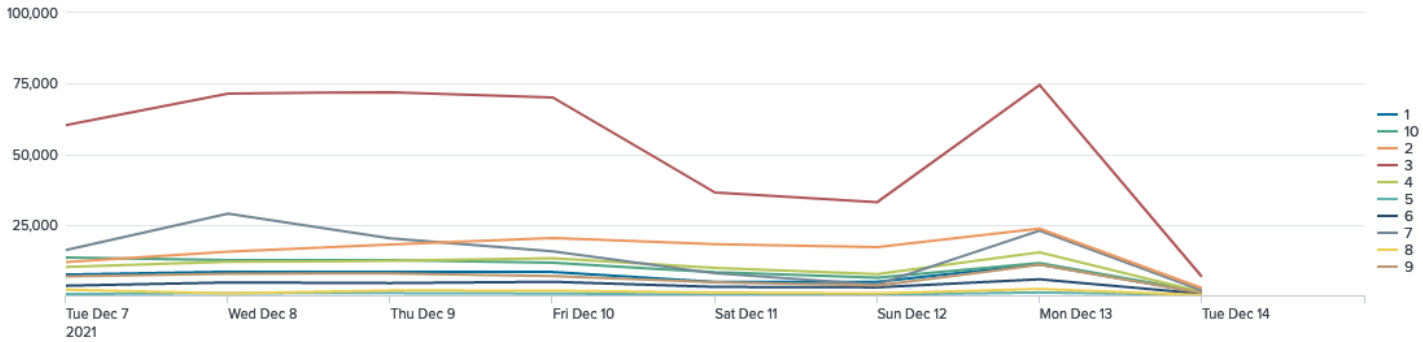
Indexed Data

```
eventtype="vmware_cbc_alerts"
| timechart span=1d dc(id) by severity
```

Data Model

```
| `vmware_tstats` earliest(All_CBC.Alerts.create_time) as create_time
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts
  by All_CBC.cbc_event_id, All_CBC.Alerts.severity
| rename All_CBC.* as *
| rename Alerts.* as *
```

```
| rename cbc_event_id as alert_id
| eval _time = strptime(create_time,"%Y-%m-%dT%H:%M:%S.%3QZ")
| timechart span=1d dc(alert_id) by severity
```



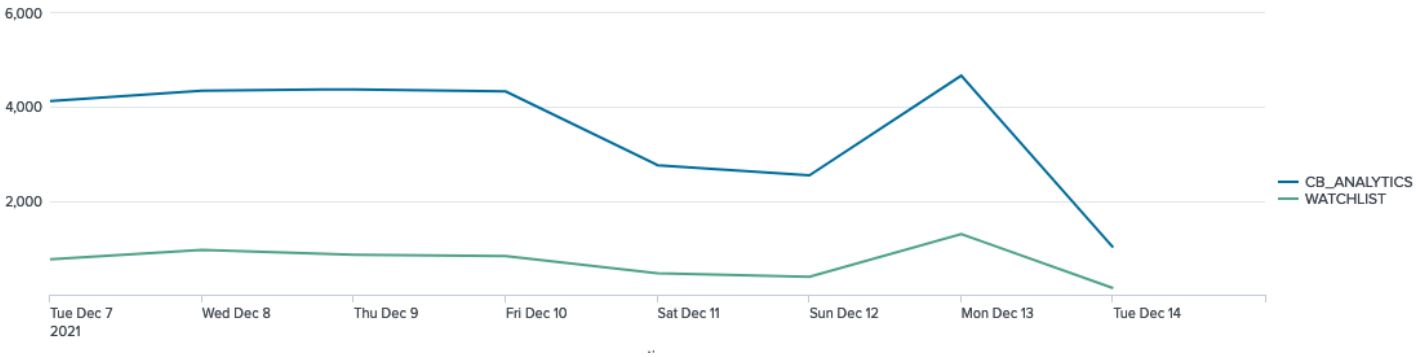
Alerted Devices over time

Indexed Data

```
eventtype="vmware_cbc_alerts"
| timechart span=1d dc(device_id) by type
```

Data Model

```
| `vmware_tstats` earliest(All_CBC.Alerts.create_time) as create_time
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts All_CBC.Alerts.severity>=1
  by All_CBC.cbc_event_id, All_CBC.device_id, All_CBC.type
| rename All_CBC.* as *
| rename Alerts.* as *
| rename cbc_event_id as alert_id
| eval _time = strptime(create_time,"%Y-%m-%dT%H:%M:%S.%3QZ")
| timechart span=1d dc(device_id) by type
```



Top Alerted Processes

Indexed Data

```
| stats
  dc(id) as alert_count,
  dc(device_id) as device_count
  by process_name
| sort -alert_count
| head 10
```

Data Model

```
| `vmware_tstats`
  dc(All_CBC.cbc_event_id) as alert_count,
  dc(All_CBC.device_id) as device_count
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts All_CBC.Alerts.severity>=1
  by All_CBC.Alerts.process_name
| rename All_CBC.* as *
| rename Alerts.* as *
| sort -alert_count
| head 10
| table process_name, alert_count, device_count
```

process_name	alert_count	device_count
wevtutil.exe	2577	1
cmd.exe	1295	3
svchost.exe	460	49
services.exe	163	2
SYSTEM	87	8
tiworker.exe	20	2
msedge.exe	18	2
googleupdate.exe	15	6
updatenotificationmgr.exe	10	3
nessus-service.exe	8	2

Top Watchlists & Reports

Products Required: Enterprise EDR

Indexed Data

```
eventtype="vmware_cbc_watchlist"
| stats
  values(watchlists{}.name) as watchlist_names,
  max(severity) as severity,
  dc(id) as alert_count,
  dc(device_id) as device_count,
  dc(process_name) as process_count,
  by report_name
| sort -alert_count
| head 10
```

Data Model

```
| `vmware_tstats`
  max(All_CBC.Alerts.severity) as severity,
  values(All_CBC.Alerts.watchlist_name) as watchlist_names,
  dc(All_CBC.cbc_event_id) as alert_count,
  dc(All_CBC.device_id) as device_count,
  dc(All_CBC.process) as process_count
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts All_CBC.Alerts.severity>=1
  by All_CBC.Alerts.report_name
| rename All_CBC.* as *
| rename Alerts.* as *
| sort -alert_count
| head 10
```

```
| table report_name, watchlist_names, severity, alert_count, device_count, process_count
```

report_name	watchlist_names	severity	alert_count	device_count	process_count
Defense Evasion - Windows Event Command Line Utility Use	Carbon Black Endpoint Suspicious Indicators	8	3869	1	2
EXE_Source ips with severity 10	Proofpoint Emerging Threats EXE_Source	10	219	49	4
EXE_Source ips with severity 9	Proofpoint Emerging Threats EXE_Source	9	129	12	2
测试1	测试	4	46	3	6
Report test	SRPTEST	5	40	5	2
report	Jess-test-nov-17-1 Jess-test-nov-17-3	5	24	3	1
SYSTEM	Jess-Test-Nov-15th	5	12	3	1
Testing	Jess-Nov-15th-test-2	5	12	3	1
EXE_Source ips with severity 8	Proofpoint Emerging Threats EXE_Source	8	9	6	2
Persistence - Regmod Run or Runonce Key Modification	Carbon Black Endpoint Suspicious Indicators	3	8	3	4

Blocked Malware

Products Required: Endpoint Standard

Indexed Data

```
eventtype="vmware_cbc_cb_analytics"
blocked_threat_category IN ("KNOWN_MALWARE", "NEW_MALWARE")
| stats
  dc(id) as alert_count,
  dc(device_id) as device_count,
  values(device_name) as device_names,
  values(sensor_action) as sensor_actions
  by process_name
| sort -alert_count
```

Data Model

```
| `vmware_tstats`
  dc(All_CBC.cbc_event_id) as alert_count,
  dc(All_CBC.device_id) as device_count,
  values(All_CBC.device_name) as device_names,
  values(All_CBC.sensor_action) as sensor_actions
  from datamodel=VMWare_CBC
  where nodename=All_CBC.Alerts All_CBC.Alerts.severity>=1 All_CBC.Alerts.blocked_threat_category IN
("NEW_MALWARE","KNOWN_MALWARE")
  by All_CBC.Alerts.process_name
| rename All_CBC.* as *
| rename Alerts.* as *
| sort -alert_count
| table process_name, alert_count, device_count, device_names, sensor_actions
```

process_name	alert_count	device_count	device_names	sensor_actions
powershell.exe	11	1	WB-Auto-QA	DENY
attack_graph.py	1	1	CBC\bd-carbonblack	DENY
svch0st.exe	1	1	DESKTOP-ISO1v1	DENY

Device Enrichment

If you're investigating an endpoint, you may want the latest information in Splunk. With the `cbcdvcinfo` command, you can bring up to date device metadata into your query. The full schema is available via the Devices API on Developer Network.

The command first needs to be configured in the app configuration under the "Custom Commands" tab. Limit your Splunk search to 100 devices to avoid potential API throttling.

Most Risky Endpoints

Required Product: Any

Required Data: Alerts (App Input or Data Forwarder), VMware CBC Device Info (App Custom Command)

This query gets device metadata, such as the last contact time, Carbon Black Cloud sensor version & state, and OS version, for the endpoints in your environment that have triggered the highest severity alerts.

Enrich Endpoint Info for risky endpoints

```
eventtype="vmware_cbc_alerts" severity >= 8
| stats dc(id) as alert_count by device_id, org_key
| sort -alert_count
| head 10
| cbcdvcinfo
| table org_key, device_id, name, alert_count, sensor_version, last_contact_time, os_version, sensor_states
```

org_key	device_id	name	alert_count	sensor_version	last_contact_time	os_version	sensor_states
ABCD1234	4467271	Carbonblack-win1	11338	3.7.0.1253	2021-12-14T03:07:32.720Z	Windows 10 x64	ACTIVE LIVE_RESPONSE_NOT_RUNNING LIVE_RESPONSE_NOT_KILLED LIVE_RESPONSE_ENABLED
ABCD1234	4483118	CBTest-2	93	3.7.0.1253	2021-12-14T03:06:17.973Z	Windows 10 x64	ACTIVE LIVE_RESPONSE_NOT_RUNNING LIVE_RESPONSE_NOT_KILLED LIVE_RESPONSE_ENABLED
ABCD1234	4483137	CBTest-1	93	3.7.0.1253	2021-12-14T03:06:52.478Z	Windows 10 x64	ACTIVE LIVE_RESPONSE_NOT_RUNNING LIVE_RESPONSE_NOT_KILLED LIVE_RESPONSE_ENABLED
ABCD1234	4857176	KG-App-VM-1	79	3.7.0.1503	2021-12-14T03:08:17.483Z	Windows 10 x64	ACTIVE

Endpoints Not Checking In

Required Product: Enterprise EDR

Required Data: Endpoint Events (Data Forwarder), VMware CBC Device Info (App Custom Command)

If an endpoint stops checking in to Carbon Black Cloud, it may warrant investigation.

This can be accomplished by forwarding process start endpoint events (custom query filter `type:endpoint.event.procstart`) to Splunk. They're only a few percent of all EDR data but provide tremendous visibility. Nearly every endpoint will generate process starts during a normal day.

Consider filtering further, such as by device group or policy, to segment:

- Critical endpoints such as Domain Controllers or production servers, where investigation should occur if data is missing for just a few hours.
- User endpoints such as laptops, where being offline for days may be normal.

Run this query across the past 5-10 days. It will identify the top 100 endpoints whose last event was received more than 24 hours ago, then retrieve the endpoint's last contact time from the Carbon Black Cloud Devices API.

Note it uses the app's built-in data model to get days' worth of event data quickly, so you'll need to run this with the VMware Carbon Black Cloud app selected. It's a long query, but most of it is just timestamp wrangling.

Endpoints not checking in

```
    | `vmware_tstats`
      latest(All_CBC.Endpoint.backend_timestamp) as last_event_data
      from datamodel=VMWare_CBC
      where nodename=All_CBC.Endpoint
      by All_CBC.device_name, All_CBC.device_id, All_CBC.org_key
    | rename All_CBC.* as *
    | eval epoch_last_event_data = strptime(last_event_data, "%Y-%m-%d %H:%M:%S %z")
    | eval window = relative_time(now(), "-24h@h")
    | where epoch_last_event_data <= window
    | sort last_event_data
    | head 100
    | cbcdvcinfo
    | eval epoch_last_contact = strptime(last_contact_time, "%Y-%m-%dT%H:%M:%S.%3N%Z")
    | eval diff = floor(abs(epoch_last_contact - epoch_last_event_data))
    | eval diff_between_last_contact_and_data = tostring(diff, "duration")
    | table device_name, device_id, org_key, last_contact_time, last_event_data, diff_between_last_contact_and_data
```

device_name	device_id	org_key	last_contact_time	last_event_data	diff_between_last_contact_and_data
Workbench-test-	4082591	ABCD1234	2021-12-11T06:55:06.523Z	2021-12-11 06:59:28 +0000 UTC	00:04:21
ATTACK\pc2	4532326	ABCD1234	2021-12-14T03:07:42.825Z	2021-12-12 18:12:43 +0000 UTC	1+08:54:59
DESKTOP-98CPI3M	4950371	ABCD1234	2021-12-13T02:52:31.523Z	2021-12-13 02:53:00 +0000 UTC	00:00:28

Threat Hunting

MITRE

Carbon Black Cloud aligns to the MITRE ATT&CK Framework in both CB Analytics Alerts and Watchlist Hits. This enables easy and effective threat hunting, such as:

- Identifying and investigating endpoints that have observed the most unique MITRE TIDs
- Identifying and investigating endpoints that have observed specific MITRE TIDs, such as those used in an emerging threat
- Identifying which MITRE TIDs have been observed on a specific endpoint already under investigation

CB Analytics Alerts

Required Product: Endpoint Standard

Required Data: Alerts (App Input or Data Forwarder)

CB Analytics Alert TTPs contain MITRE techniques. In addition to the MITRE TID, they have a concise description of the technique, which is handy for those of us who don't have the entire ATT&CK framework memorized. Any time a sub-technique's TTP is included with a CB Analytics Alert, the parent's TTP will also be included.

Examples: `MITRE_T1596_SEARCH_OPEN_TECHNICAL_DATABASES` and `MITRE_T1596_001_DNS_PASSIVE_DNS`

Carbon Black Cloud strives to keep current with the MITRE's routine updates the ATT&CK framework. Deprecated MITRE TTPs will still be included with alerts for 12 months. If you are using a specific MITRE TID for detections, keep up to date with the latest changes.

- Carbon Black Cloud's list of deprecated & supported MITRE TIDs, including recommended new TIDs for those which have been deprecated
- Carbon Black Cloud MITRE v7 deprecation announcement
- Carbon Black Cloud Developer Newsletter (so you're notified of any planned deprecations with plenty of notice)

Top Endpoints by unique MITRE TID count

```
eventtype="vmware_cbc_cb_analytics"
| rename threat_indicators{}.ttps{} as mitre_ttps
| mvexpand mitre_ttps
| rex field=mitre_ttps "^MITRE_(?<mitre_tid>T\d+(\_\d{3})?)_(?<mitre_tid_name>.*)$"
| where not isnull(mitre_tid)
| eval mitre_tid = lower(mitre_tid)
| stats
  dc(mitre_tid) as mitre_tid_count
  values(mitre_tid) as mitre_tids,
  values(mitre_tid_name) as mitre_tid_names
  by device_id, device_name
| sort -mitre_tid_count
```

Top MITRE TIDs by endpoint count

```
eventtype="vmware_cbc_cb_analytics"
| rename threat_indicators{}.ttps{} as mitre_ttps
| mvexpand mitre_ttps
| rex field=mitre_ttps "^MITRE_(?<mitre_tid>T\d+(\_\d{3})?)_(?<mitre_tid_name>.*)$"
| where not isnull(mitre_tid)
| eval mitre_tid = lower(mitre_tid)
| stats
  dc(device_id) as device_count,
  values(device_name) as device_names,
  by mitre_tid, mitre_tid_name
| sort -device_count
```

device_id	device_name	mitre_tid_count	mitre_tids	mitre_tid_names
4885462	QA\PSBSYS	4	t1055 t1057 t1059 t1106	CMD_LINE_OR_SCRIPT_INTER NATIVE_API PROCESS_DISCOVERY PROCESS_INJECT
4873393	QA\782021w2k16	3	t1046 t1057 t1059	CMD_LINE_OR_SCRIPT_INTER NETWORK_SERVICE_SCANNING PROCESS_DISCOVERY
4938802	DESKTOP-E3NJVI9	3	t1046 t1057 t1543	CREATE_OR_MODIFY_SYS_PROC NETWORK_SERVICE_SCANNING PROCESS_DISCOVERY

mitre_tid	mitre_tid_name	device_count	device_names
t1046	NETWORK_SERVICE_SCANNING	7	DESKTOP-E3NJVI9 QA\782021w2k16 SUPLAB-433-WIN WIN10-LCHAI manticorewin832 win10-CBClient1 win10-ps-moid
t1057	PROCESS_DISCOVERY	7	Brian-Thrashbx2 CBSTD-WKSH DESKTOP-E3NJVI9 PSB760 QA\782021w2k16 QA\GL-PI-SRP-79 QA\PSBSYS
t1059	CMD_LINE_OR_SCRIPT_INTER	5	Brian-Thrashbx2

Watchlist Hits

Required Product: Enterprise EDR

Required Data: Watchlist Hits (Data Forwarder)

Watchlist hits also contain MITRE techniques in the report tags. While the tag doesn't contain the technique name, we can substitute the report name.

Top Endpoints by unique MITRE TID count

```
eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:s3:watchlist:hits"
| rename report_tags{} as mitre_tids
| mvexpand mitre_tids
| regex mitre_tids="^t\d+$"
| eval mitre_tid_name = report_name
| stats
  dc(mitre_tids) as mitre_tid_count,
  values(mitre_tids) as mitre_tids,
  values(mitre_tid_name) as mitre_tid_names
  by device_id, device_name
| sort -mitre_tid_count
```

Top MITRE TIDs by endpoint count

```
eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:s3:watchlist:hits"
| rename report_tags{} as mitre_tids
| mvexpand mitre_tids
| regex mitre_tids="^t\d+$"
| eval mitre_tid_name = report_name
```

```
| stats
  values(mitre_tid_name) as mitre_tid_names,
  dc(device_id) as device_count,
  values(device_name) as device_names,
  by mitre_tids
| sort -device_count
```

device_id	device_name	mitre_tid_count	mitre_tids	mitre_tid_names
4833716	OR-EP01-VN	14	t1016 t1021 t1037 t1047 t1049 t1059 t1070 t1082 t1136 t1175 t1543 t1547 t1552 t1557	Credential Access - Credentials in Files #2 Credential Access - LLMNR/NBT-NS Poisoning - LLMNR Traffic Detected Defense Evasion - File Deletion Discovery - System Information Discovery Discovery - System Network Configuration Discovery #5 Execution - Command and Scripting Interpreter Execution Execution - System Profiling via WMI Lateral Movement - DCOM - svchost Launching Command Interpreter Lateral Movement - Remote Desktop Protcol Login Detected Lateral Movement - Remote Desktop Protocol Lateral Movement - Remote Desktop Protocol #2 Persistence - Create Accounts Using GUI Persistence - Regmod Run or Runonce Key Modification Persistence - Service Deleted via sc.exe
4938665	DESKTOP-U8Q2O2M	14	t1007 t1012 t1016 t1021 t1037 t1049 t1059	Credential Access - LLMNR/NBT-NS Poisoning - LLMNR Traffic Detected Defense Evasion - BITS Jobs - BitsAdmin Policy Modification Discovery - Query Registry Discovery - System Information Discovery Discovery - System Network Connections Discovery #2 Discovery - System Service Discovery - net.exe Start

mitre_tids	mitre_tid_names	device_count	device_names
t1557	Credential Access - LLMNR/NBT-NS Poisoning - LLMNR Traffic Detected	50	2748S-W10-CB1 ATTACK\pc2 AWS-WEBSERVER BAS\bas-carbonblack Brian-Thrashbx2 CB-H10 Carbonblack-win1 DEMO\HR-SERVER-EAST DESK-F1-178 DESKTOP-98CPI3M DESKTOP-E3NJVI9 DESKTOP-L3LAAMH DESKTOP-U8Q2O2M EC2AMAZ-9TMIHI7 EXAPIL\PIL-CB7-2 MARVEL\win10-abdullah OESISREMOTE\OR-EP01-VN

Combining CB Analytics Alerts & Watchlist Hits

Required Product: Endpoint Standard & Enterprise EDR

Required Data: Alerts (App Input or Data Forwarder), Watchlist Hits (Data Forwarder)

If you have both Endpoint Standard and Enterprise EDR, these queries can be combined into a unified view of MITRE.

Top MITRE TIDs by endpoint count, including the technique name and data source.

```
eventtype="vmware_cbc_cb_analytics"
```

```
| rename threat_indicators{}.ttps{} as mitre_ttps
| mvexpand mitre_ttps
| rex field=mitre_ttps "^MITRE_(?<mitre_tid>T\d+(\_\d{3})?)_(?<mitre_tid_name>.*)$"
| where not isnull(mitre_tid)
| eval mitre_tid = lower(mitre_tid), source = "CB Analytics Alerts"
| fields device_id, device_name, mitre_tid, mitre_tid_name, source
| append [
  search eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:s3:watchlist:hits"
  | rename report_tags{} as mitre_tids
  | mvexpand mitre_tids
  | regex mitre_tids="^t\d+$"
  | eval mitre_tid_name = report_name, source = "Watchlist Hits"
  | fields device_id, device_name, mitre_tid, mitre_tid_name, source
]
| stats
  dc(mitre_tid) as mitre_tid_count
  values(mitre_tid) as mitre_tids,
  values(mitre_tid_name) as mitre_tid_names,
  values(source) as sources
  by device_id, device_name
| sort -mitre_tid_count
```

device_id	device_name	mitre_tid_count	mitre_tids	mitre_tid_names	sources
4885462	QA\PSBSYS	4	t1055 t1057 t1059 t1106	CMD_LINE_OR_SCRIPT_INTER Credential Access - LLMNR/NBT-NS Poisoning - LLMNR Traffic Detected Discovery - Query Registry Discovery - System Information Discovery Discovery - System Service Discovery Detected Execution - Command and Scripting Interpreter - Powershell Execution - Command and Scripting Interpreter Execution Execution - Command-Line Interface (cmd.exe /c) Execution - Powershell Execution With Unrestricted or Bypass Flags Detected Lateral Movement - DCOM - svchost Launching Command Interpreter Lateral Movement - Remote Desktop Protocol NATIVE_API PROCESS_DISCOVERY PROCESS_INJECT	CB Analytics Alerts Watchlist Hits

Commonly Abused Commands

Required Product: Enterprise EDR

Required Data: Endpoint Events (Data Forwarder)

The blog Windows Commands Abused by Attackers is a great reference to help identify early stages of initial investigation and reconnaissance. Individually, these are normal commands. However, when multiple are observed on the same endpoint in a short time period, it may be cause for investigation.

This can be accomplished by forwarding process start endpoint events (custom query filter `type:endpoint.event.procstart`) to Splunk.

This query uses a long regex to pull out these commands, only when they're a standalone word. You may want to do additional tuning such as allowing adjacent characters like `.` and `/` as well as allow-listing certain known good processes that invoke some of these commands and might contribute to noise.

Endpoints with the most commonly abused commands

```
| `vmware_tstats` count from datamodel=VMWare_CBC where nodename=All_CBC.Endpoint by All_CBC.device_name,
All_CBC.device_id, All_CBC.org_key, All_CBC.Endpoint.process_cmdline, All_CBC.process_executable
| rename All_CBC.* as *
| rename Endpoint.process_cmdline as cmdline
| rex field=cmdline
"(^|\s)(?<command>tasklist|ver|ipconfig|systeminfo|net\stime|netstat|whoami|net\sstart|qprocess|query|dir|net\s
view|ping|net\suse|type|net\suser|net\slocalgroup|net\sgroup|net\sconfig|net\sshare|at|reg|wmic|netsh\sadvfirew
all|sc|wusa)($|\s)"
| where not isnull(command)
| stats dc(command) as command_count, values(command) as commands, values(process_executable) as processes,
values(cmdline) as cmdlines by device_name, device_id
```

```
    | sort -command_count
```

device_name	device_id	command_count	commands	processes	cmdlines
OR-EP01-VN	4833716	5	ipconfig ping sc ver wmic	c:\program files\confer\blades \livequery\osqueryi.exe c:\windows\system32\cmd.exe c:\windows\system32\ipconfig.exe c:\windows\system32\ping.exe c:\windows\system32\sc.exe c:\windows\syswow64\cmd.exe c:\windows\syswow64 \wbem\wmic.exe	"C:\Program Files\Confer\Blades\LiveQuery\osqueryi.exe" --flagfile="C:\Progr cast(trim(substr(filename, instr(filename, '.')+1), '.jar') as REAL) as ver, and ver < 15;" "cmd.exe" /c wmic bios get serialnumber "cmd.exe" /c wmic computersystem get model C:\Windows\system32\cmd.exe /c ipconfig > ip.txt ipconfig ipconfig ipconfig /release ipconfig /renew ping oesisremote.test ping or-ep01.oesisremote.test sc del WaRemoteTest sc delete WaRemoteTest wmic bios get serialnumber wmic computersystem get model
BAS\bas-carbonblack	4242869	4	netstat ping systeminfo wmic	c:\windows\syswow64\cmd.exe c:\windows\syswow64\netstat.exe c:\windows\syswow64\ping.exe c:\windows\syswow64 \systeminfo.exe c:\windows\syswow64 \wbem\wmic.exe	C:\Windows\system32\cmd.exe /c systeminfo /FO CSV && wmic CPU get Name /FORM C:\Windows\system32\cmd.exe /c wmic computersystem get Name, UserName /FORMA C:\Windows\system32\cmd.exe /c wmic diskdrive where "MediaType='Removable Me C:\Windows\system32\cmd.exe /c wmic path CIM_LogicalDevice where "Descriptio /FORMAT:LIST C:\Windows\system32\cmd.exe /c wmic process get Name, ProcessId, HandleCount netstat -anp tcp ping -n 4 -w 4000 8.8.8.8

Log4Shell

Required Product: Any

Required Data: Audit Logs (App Input)

You can leverage the Data Forwarder and Custom Query Filters to forward Log4Shell-relevant EDR data to Splunk. Check out Tech Zone article for Detecting Log4j in the Carbon Black Console to learn more.

Identify java spawning powershell

```
    | tstats summariesonly=t
      values(All_CBC.Endpoint.parent_process_exec) AS parent_process_cmdline,
      values(All_CBC.Endpoint.procstart.childproc_name) as childproc_names,
      values(All_CBC.process_guid) as process_guids,
      values(All_CBC.type) as event_types
      latest(_time) AS latest,
      earliest(_time) AS earliest
      from datamodel=VMWare_CBC
      where ((All_CBC.Endpoint.parent_process_exec="*java*" OR All_CBC.Endpoint.parent_process_exec="*tomcat*") AND
    All_CBC.process_executable="*powershell*")
      by All_CBC.device_name, All_CBC.device_id, All_CBC.org_key, All_CBC.process_executable
    | rename All_CBC.* as *, process_executable as process_cmdline
    | eval _time=latest
    | reltime
    | convert ctime(latest), ctime(earliest)
    | table device_id, device_name, event_types, process_guids, parent_process_cmdline, process_cmdline,
    childproc_names, reltime, earliest, latest
```

device_id	device_name	event_types	process_guids	parent_process_cmdline	process_cmdline
3945000	DESKTOP- E33JRXY	endpoint.event.procstart	7YQSBCDE-003c325c- 000036c8-00000000-1d7f6ff79a9e054	d:\java\jdk1.8.0_231 \bin\java.exe	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -version 2 -NoProfile -EncodedCommand JgAgAHsACgBbAEMAbwBuAHMAbwBsAGUAXQA6ADoATwB1AHQAcAB1AHQARQBuAGMAbwBkAGkAbgBnACAAPQAgAFsAUwB5AHMAdAB1AG0ALgBUAGUAeAE

Audit Logs

Required Product: Any

Required Data: Audit Logs (App Input)

Audit Logs have visibility into everything happening in your Carbon Black Cloud environment. Using Splunk's built-in iplocation command can further enhance the audit data to help quickly identify users logging in from unexpected locations.

Flagged Audit Logs

Carbon Black Cloud flags suspicious audit logs, such as when failed logins come from previously unused IP addresses and if an account is locked due to too many failed login attempts.

Flagged Audit Logs

```
eventtype="vmware_cbc_auditlogs" flagged=true
| iplocation src
| eval location = case(Region = "", Country, City = "", Region + ", " + Country, 1=1, City + ", " + Region + ",
" + Country)
| table _time, user, src, location, description
| sort -_time
```

Logins

Sometimes, it's helpful to go back to the basics and monitor login successes and failures.

List login successes and failures

```
eventtype="vmware_cbc_auditlogs" ("Logged In" OR "Login") NOT "Connector"
| iplocation src
| eval success = if(description = "Logged in successfully" OR like(description, "Login in successfully through
%"), "Login Success", "Login Failure")
| eval location = case(Region = "", Country, City = "", Region + ", " + Country, 1=1, City + ", " + Region + ",
" + Country)
| table _time, user, src, location, success
| sort -_time
```

_time	user	src	location	success
2021-12-14 09:37:31	bdeakyne@carbonblack.com	73.217.0.4	Boulder, Colorado, United States	Login Success
2021-12-14 09:37:25	bdeakyne@carbonblack.com	73.217.0.4	Boulder, Colorado, United States	Login Failure
2021-12-14 09:17:52	user2@partner2.co.uk	79.177.0.229	Ramat Gan, Tel Aviv, Israel	Login Success
2021-12-14 06:36:13	user1@partner1.com	1.39.0.10	Bhubaneswar, Odisha, India	Login Success
2021-12-14 05:20:55	user2@partner2.co.uk	79.177.0.229	Ramat Gan, Tel Aviv, Israel	Login Success
2021-12-14 04:52:51	user1@partner1.com	1.39.0.20	Bhubaneswar, Odisha, India	Login Success
2021-12-14 03:59:35	user3@vmware.com	122.172.0.28	Bengaluru, Karnataka, India	Login Success

Live Response

Live Response is a powerful capability that provides remote access to endpoints. All Live Response sessions and commands should be closely monitored; pay special attention to the `create process`, `kill`, and `put file` commands. Some SOCs even fire an alert whenever a Live Response session is initiated.

This search lists each endpoint and user combination, along with the actions, commands, and details of each session.

Audit Live Response details by user and endpoint

```
eventtype="vmware_cbc_auditlogs" description="*liveresponse*"
| eval description = replace(description, "[\r\n]","")
| rex field=description "LiveResponse\s+(?<device_name>[^\|]+)\|Action\s+(?<action>[^\|]+)\|URL\s+/live\-
response\?deviceID=(?<device_id>\d+)\|(Details\s+(?<details>.*))?"
| rex field=details "\"name\": \"(?<command>[^\"]+)\""
| iplocation src
| eval location = case(Region = "", Country, City = "", Region + ", " + Country, 1=1, City + ", " + Region + ",
" + Country)
| stats
  values(action) as actions,
  values(command) as commands,
  values(details) as details,
  values(location) as locations,
  by device_id, device_name, src, location, user
```

device_id	device_name	src	location	user	actions	commands	details	locations
3925348	carbon-black-integration-endpoint	54.212.0.210	Boardman, Oregon, United States	user1@partner1.com	Command Created File Created File Retrieved via Download Url	create process delete file directory list get file kill memdump process list put file reg create key reg delete key reg delete value reg enum key reg set value	Command(0): {"name": "reg enum key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test", "session_id": "1105:3925348"} Command(1): {"name": "delete file", "path": "C:\\test\\test.txt", "session_id": "1105:3925348"} Command(10): {"file_id": "c37b6ea8-2fb8-4632-a03e-3989502d1137", "name": "put file", "path": "C:\\test\\test.txt", "session_id": "1105:3925348"} Command(11): {"name": "reg create key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test\\test_sub_key", "session_id": "1105:3925348"} Command(12): {"name": "get file", "path": "c:\\windows\\temp\\cblr.kFyLiY0MyoCQ.tmp", "session_id": "1105:3925348"} Command(13): {"name": "get file", "path": "C:\\test\\test.txt", "session_id": "1105:3925348"} Command(14): {"name": "reg enum key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test", "session_id": "1105:3925348"} Command(15): {"name": "delete file", "path": "c:\\windows\\temp\\cblr.kFyLiY0MyoCQ.tmp", "session_id": "1105:3925348"} Command(16): {"name": "directory list", "path": "C:\\test\\memdump_test\\", "session_id": "1105:3925348"} Command(17): {"name": "kill", "pid": 4744, "session_id": "1105:3925348"} Command(18): {"name": "delete file", "path": "C:\\test\\test.txt", "session_id": "1105:3925348"} Command(19): {"name": "reg enum key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test", "session_id": "1105:3925348"} Command(2): {"name": "process list", "session_id": "1105:3925348"} Command(20): {"name": "create process", "output_file": "c:\\windows\\temp\\cblr.ZHEdYjZnQRoL.tmp", "path": "cmd.exe /c C:\\test\\get_pid.bat", "session_id": "1105:3925348", "wait": true} Command(21): {"name": "get file", "path": "c:\\windows\\temp\\cblr.ZHEdYjZnQRoL.tmp", "session_id": "1105:3925348"} Command(22): {"name": "reg delete value", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test\\test_val", "session_id": "1105:3925348"} Command(23): {"name": "delete file", "path": "c:\\windows\\temp\\cblr.ZHEdYjZnQRoL.tmp", "session_id": "1105:3925348"} Command(24): {"name": "reg delete key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test\\test_sub_key", "session_id": "1105:3925348"} Command(25): {"name": "reg delete key", "path": "HKEY_LOCAL_MACHINE\\SOFTWARE\\test", "session_id": "1105:3925348"}	Boardman, Oregon, United States

User Activity

It can be helpful to break out user activity by category to see at a glance what each user (or API key) is doing and where from. Since most of the valuable information in the audit log is in plain text, the query parses the most common audit logs and outputs a category for more effective grouping. This is by no means a complete list, but should capture many common scenarios.

Summarize user activity

```
eventtype="vmware_cbc_auditlogs" user=*
| eval category = case(
like(description, "Updated report %"), "Report Updated",
like(description, "Connector % logged in successfully"), "API Key Connection",
like(description, "A bulk policy update has been applied%"), "Device Policy Updated",
like(description, "Changed device % policy %"), "Device Policy Updated",
like(description, "Added API ID %"), "API Key Created",
like(description, "Added user %"), "User Created",
like(description, "Alert % dismissed %"), "Alert Dismissed",
like(description, "Changed Password%"), "User Password Changed",
like(description, "Created Config: {%"), "Data Forwarder Created",
like(description, "Created custom role %"), "Role Created",
like(description, "Custom role % updated %"), "Role Updated",
like(description, "Deleted API ID %"), "API Key Deleted",
like(description, "Deleted Config: %"), "Data Forwarder Deleted",
like(description, "Device % uninstalled"), "Device Uninstalled",
like(description, "Downloaded policy configuration%"), "Policy Downloaded",
like(description, "LiveResponse%"), "Live Response",
like(description, "Logged in successfully"), "User Login Success",
like(description, "Login in successfully through %"), "User Login Success (MFA)",
like(description, "Password did not match %"), "User Login Failure",
like(description, "Password reset %"), "User Password Reset",
like(description, "Policy % was deleted"), "Policy Deleted",
like(description, "Account locked %"), "User Account Locked",
like(description, "Policy % was modified%"), "Policy Modified",
like(description, "Ran query%"), "Live Query Executed",
like(description, "Created grant%"), "User Grant Created",
like(description, "Requested sensor upgrade%"), "Device Version Update Requested",
```

```
like(description, "Searched for reputations%"), "Reputation Search",
like(description, "Sensor Bypass Enabled%"), "Sensor Bypass Enabled",
like(description, "Sensor Bypass Disabled%"), "Sensor Bypass Disabled",
like(description, "Set BACKGROUND_SCAN to off%"), "Device Background Scan Disabled",
like(description, "Set BACKGROUND_SCAN to on%"), "Device Background Scan Enabled",
like(description, "Set BYPASS to off%"), "Device Un-Bypassed",
like(description, "Set BYPASS to on%"), "Device Bypassed",
like(description, "Set QUARANTINE to off%"), "Device Un-Quarantine",
like(description, "Set QUARANTINE to on%"), "Device Quarantined",
like(description, "Successfully exported query results%"), "Live Query Results Exported",
like(description, "%Policy Settings Changed%"), "Policy Updated",
like(description, "Updated Config:%"), "Data Forwarder Updated",
like(description, "Successfully confirmed the email%"), "User Confirmed Email",
like(description, "Updated custom role%"), "Role Updated",
like(description, "Updated watchlist%"), "Watchlist Updated",
like(description, "User % added Reputation Override%"), "Reputation Override Added",
like(description, "User % retrieved secret%"), "API Key Secret Viewed",
like(description, "User % accepted EULA%"), "User Accepted EULA",
like(description, "%Tab:Settings (Enrollment)%"), "Viewed Devices Page",
like(description, "Hash % was %requested to be deleted %"), "Device Malware Deleted",
like(description, "Failure deleting hash %"), "Device Malware Delete Failed",
like(description, "Looked up reputation %"), "Reputation Lookup",
like(description, "Replaced % reports in %"), "Watchlist Updated",
like(description, "Re-registration of device %"), "Device Re-Registered",
like(description, "Sent deregister request %"), "Device De-Registered",
like(description, "Delete Deregistered Devices %"), "Device Deleted",
like(description, "% was deleted from your organization %device%"), "Device Deleted",
like(description, "% deregistered through %"), "Device De-Registered",
like(description, "Success deleting hash % off of device %"), "Device Malware Deleted",
like(description, "Downloaded output of job %"), "Search Results Downloaded",
like(description, "Sensor registered on VM %"), "Device Registered on VM",
like(description, "Invalid authentication method %"), "User Login Failure",
like(description, "Upload Hash %"), "Device Malware Upload",
like(description, "%Tab:Malware Removal%"), "Viewed Malware Removal Page",
like(description, "Delete Hash % requested for device%"), "Device Malware Deleted",
like(description, "Going to update device% to version%"), "Device Version Updated",
like(description, "Created event export job%"), "Search Results Exported",
like(description, "Registration Id not found for VM %"), "Device Registration on VM Failure",
like(description, "Request to delete owner %"), "User (Owner) Deleted",
like(description, "Sent upgrade sensor version request %"), "Device Version Update Requested",
like(description, "%devices were bulk % policy%"), "Device Policy Updated",
like(description, "Bulk ignored % report%"), "Watchlist Report Ignored",
like(description, "Appliance%Registration %"), "Appliance Registered",
like(description, "Success"), "Something was Successful",
like(description, "Uploaded file%"), "File Uploaded",
like(description, "Log export % for appliance %"), "Appliance Logs Exported",
like(description, "Updated access profile %"), "Role Updated",
like(description, "Updated user %"), "User Updated",
like(description, "Alert % undismissed %"), "Alert Undismissed",
like(description, "Undismissed % alert%"), "Alert Undismissed",
like(description, "Dismissed % alert%"), "Alert Dismissed",
like(description, "Created policy%"), "Policy Created",
like(description, "Created%feed%"), "Feed Created",
like(description, "Created%watchlist%"), "Watchlist Created",
like(description, "Created report%"), "Watchlist Report Created",
like(description, "Deleted custom role%"), "Role Deleted",
like(description, "Deleted grant%"), "User Grant Deleted",
like(description, "Initiated request%to undismiss%"), "Alert Undismissed",
like(description, "Initiated request%to dismiss%"), "Alert Dismissed",
like(description, "Query%deleted%"), "Live Query Deleted",
like(description, "Query run%stopped%"), "Live Query Run Stopped",
like(description, "%dashboard%"), "Dashboard Updated",
like(description, "Set Quarantine to On%"), "Device Quarantined",
like(description, "Set Quarantine to Off%"), "Device Unquarantined",
like(description, "Set Bypass to On%"), "Device Bypassed",
like(description, "Set Bypass to Off%"), "Device Unbypassed",
like(description, "Updated alert notification%"), "Alert Notification Updated",
like(description, "%tagging for watchlist%"), "Watchlist Tagging Updated",
like(description, "Installation triggered %"), "Device Installation Triggered",
like(description, "Sensor installation failed%"), "Device Installation Failed",
```

```
like(description, "Regenerated API key %"), "API Key Updated",
like(description, "Updated API %"), "API Key Updated",
like(description, "Threat ID % undismissed%"), "Threat Undismissed",
like(description, "Threat ID % dismissed%"), "Threat Dismissed",
like(description, "%deleted Reputation Overrides%"), "Reputation Override Deleted",
like(description, "Vulnerability%export%"), "Vulnerabilities Exported",
like(description, "Password is not set%"), "User Password Not Set",
like(description, "Bulk-applied % policy changes %"), "Device Policy Updated",
like(description, "Deleted admin%"), "User (Admin) Deleted",
like(description, "Sensor installation is ERROR%"), "Device Sensor Installation Failed",
like(description, "Sensor installation is SUCCESS%"), "Device Sensor Installation Succeeded",
like(description, "Updated policy%"), "Policy Updated",
like(description, "Updated grant%"), "User Grant Updated",
match(description, "device"), "Device Other",
match(description, "disabled"), "Other: Disabled",
match(description, "enabled"), "Other: Enabled",
match(description, "updated"), "Other: Updated",
match(description, "changed"), "Other: Changed",
match(description, "deleted"), "Other: Deleted",
match(description, "downloaded"), "Other: Downloaded",
1=1, "Other"
)
| eval user_type = if(match(user, "^[0-9A-Z]{10}$"), "API Key", "User")
| iplocation src
| eval location = case(Region = "", Country, City = "", Region + ", " + Country, 1=1, City + ", " + Region + ",
" + Country)
| stats
  values(category) as user_actions,
  dc(Country) as country_count
  values(src) as ips,
  values(location) as locations,
  by user, user_type
```

user	user_type	user_actions	country_count	ips	locations
ABCDE12345	API Key	API Key Connection Device Quarantined Device Un-Quarantine	3	134.238.0.223 3.0.0.159 59.88.0.108 59.88.0.11	Amsterdam, North Holland, Netherlands Korba, Chhattisgarh, India Singapore
user4@vmware.com	User	User Accepted EULA User Confirmed Email User Login Success User Password Reset	2	40.94.0.16 88.203.0.110	San Antonio, Texas, United States Sofia, Sofia-Capital, Bulgaria

Vulnerabilities

Required Product: Vulnerability Management for Workloads or Endpoints

Required Data: Vulnerabilities (App Input)

Carbon Black Cloud can provide a vulnerability assessment of all workloads or endpoints. Through a partnership with Kenna Security, the assessment goes beyond traditional CVSS scores to include risk-based prioritization so your organization can focus on resolving vulnerabilities most likely to result in compromise.

The Carbon Black Cloud Splunk App vulnerabilities input queries the Get Vulnerability List API and returns the Device Vulnerability List schema.

Endpoints with Critical Vulnerabilities

A good starting point for vulnerability assessment is to identify any endpoint with critical vulnerabilities. With that information, you could move the impacted endpoints to a more restrictive policy, or just keep a closer eye on any alert that fires from that endpoint.

Endpoints with critical vulnerabilities

```
eventtype="vmware_cbc_vulnerability_os_list"severity="CRITICAL"
| rename affected_assets{} as device_name
| mvexpand device_name
| stats
  dc(cve) as critical_cve_count,
  values(cve) as critical_cves
  by device_name
| sort -critical_cve_count
```

device_name	critical_cve_count	critical_cves
Demo-windows-2	7	CVE-2017-0199 CVE-2017-8464 CVE-2020-1350 CVE-2020-1472 CVE-2021-1675 CVE-2021-34527 CVE-2021-40444

Endpoint Vulnerability Trends

Is your organization effectively mitigating vulnerabilities? Trending the number of vulnerable endpoints over the past few weeks can show that at a glance.

Vulnerability Trends

```
eventtype="vmware_cbc_vulnerability_os_list" vuln_info.risk_meter_score >= 5
| timechart span=1d dc(affected_assets{}) as device_count by severity
```

Endpoint Vulnerabilities

When investigating an endpoint during threat hunting or incident response, having a snapshot of vulnerabilities can help inform next steps. For example, if behavior indicates one of those vulnerabilities is being actively exploited, the endpoint can be immediately quarantined.

Endpoints vulnerable to each CVE

```
eventtype="vmware_cbc_vulnerability_os_list"
```

```
| rename affected_assets{} as device_name
| mvexpand device_name
| where device_name = "your-device-name-here"
| stats
  max(vuln_info.risk_meter_score) as risk_meter_score,
  values(category) as category,
  max(vuln_info.easily_exploitable) as easily_exploitable,
  max(eval(strftime(_time, "%Y-%m-%d"))) as last_seen,
  values(vuln_info.cve_description) as description,
  values(eval(mvappend(vuln_info.solution, vuln_info.fixed_by))) as solution,
  values(vuln_info.nvd_link) as link
  by cve
| sort -risk_meter_score
```

Live Query

Required Products: Audit & Remediation

Required Data: Live Query Input (App Input)

Live Query leverages osquery to interrogate endpoints. You can find the full list of tables & schemas in the osquery's site.

It's worth noting results will only be pulled into Splunk once the query completes. Scheduled queries will be marked as completed as soon as the next scheduled one begins, so if you're running daily, your data could be delayed up to a day.

Logged in users

Live Query can pull the list of logged-in users from one or more endpoints daily or on-demand from the logged_in_users table. My query is named "List Logged In Users" and does a simple osquery, schedule daily:

```
SELECT * FROM logged_in_users
```

Which endpoints does each user log into?

```
eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:livequery:result" query_name="List Logged In Users"
status=matched
| rename device.* as device_*, fields.* as *
| eval _time = time
| reltime
| stats
  values(device_name) as device_names,
  values(device_id) as device_ids
  by user
```

What users are logging into each endpoint?

```
eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:livequery:result" query_name="List Logged In Users"
status=matched
| rename device.* as device_*, fields.* as *
| eval _time = time
| reltime
| stats
  values(user) as users
  by device_name, device_id
```

user	device_names	device_ids
bdeakyne	Desktop-CB	4886724
root	SERVER-265	4322377
runlevel	SERVER-265	4322377
sarcher	SERVER-265	4322377

device_name	device_id	users
Desktop-CB	4886724	bdeakyne
SERVER-265	4322377	root runlevel sarcher

Credential Harvesting

When Credential Harvesting/Dumping is observed, analysts likely want to know which users were logged in at the time and may have had their credentials compromised.

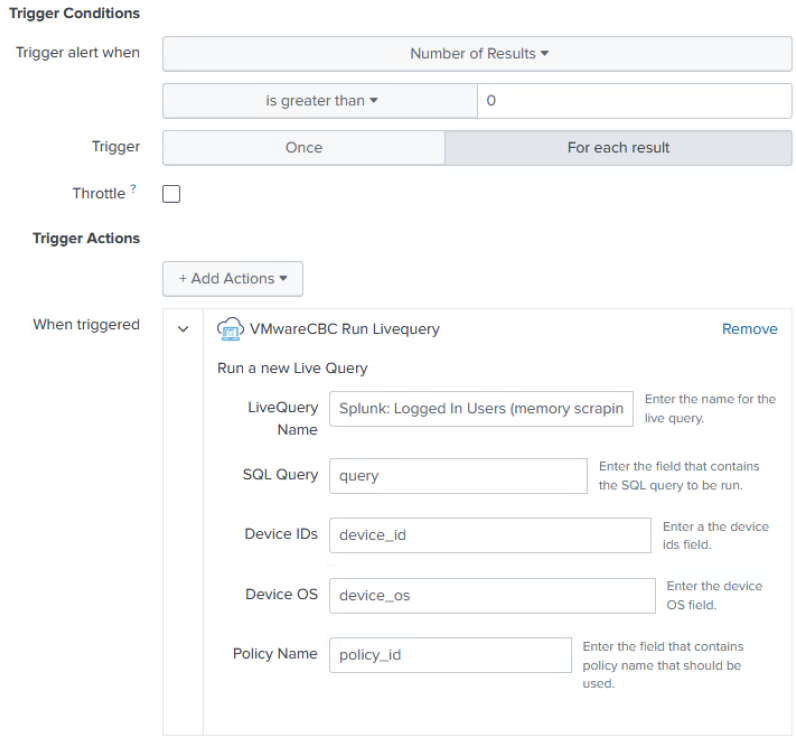
The Splunk app includes a “Run Live Query” Alert Action.

The first search will run as an Alert in Splunk on a regular basis, maybe every 5 – 10 minutes, and identify any possible instances of credential theft. It also defines the Live Query SQL we’ll run.

Query for Splunk Alert

```
eventtype="vmware_cbc_cb_analytics" threat_indicators{}.ttps{} = "MITRE_T1003_OS_CREDENTIAL_DUMP"
| eval query_logged_in_users = "SELECT * FROM logged_in_users;"
| dedup device_id
```

For each result, trigger the “VMware CBC Run Livequery” Alert Action with the parameters shown in the image below.



Now when Carbon Black Cloud detects credential scraping, Splunk will automatically figure out who was logged in to the impacted endpoints and bring in those results through the App/IA's built-in Live Query input. We can then run a splunk query to combine the original alerts with the logged in user information.

Combine Alerts & Logged In Users

```
eventtype="vmware_cbc_cb_analytics" threat_indicators{}.ttps{} = "MITRE_T1003_OS_CREDENTIAL_DUMP"
| dedup id
| reltime
| join device_id type=outer [
  search eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:livequery:result" query_name="Splunk: Logged
In Users (memory scraping)" status=matched
  | rename device.* as device_*, fields.* as *
  | eval _time = time
  | reltime
  | strcat "User=" user ", type=" tty ", SID=" sid ", " reltime login_string
  | stats values(login_string) as logins by device_id
]
| table device_id, device_name, reltime, sensor_action, threat_cause_actor_name, reason, logins
```

device_id	device_name	reltime	sensor_action	threat_cause_actor_name	reason	logins
4483137	CBTest-1	23 hours ago	TERMINATE	c:\temp\scenario \tmp\procdump.exe	The application procdump.exe attempted to create a memory dump for a system security process (lsass.exe). A Terminate action was applied.	User=bdeakyne, type=RDP-Tcp#1, SID=S-1-5-21-1183524871-323597517-2965227890-500, 1 day ago

Chrome Extensions

Live Query can fetch a full list of Chrome Extensions on a daily basis to help you inventory out-of-date or malicious add-ons. For this query, it's probably more interesting to look at the less popular extensions that appear on only a few endpoints.

Note: due to the 10,000 result limit per query, you may want to tune your Live Query SQL to filter out known-good extensions.

Show the least-used Chrome Extensions, versions, and associated endpoints/users

```
eventtype="vmware_cbc_base_index" sourcetype="vmware:cbc:livequery:result" query_name="Chrome Extensions"
status=matched
| rename device.* as device_*, fields.* as *
| stats
  values(version) as versions,
  dc(device_id) as device_count,
  values(device_name) as devices,
  values(username) as users,
  by name
| sort device_count
```

name	versions	device_count	devices	users
Google Slides	0.9	1	SDE\USBO1SEFS-01	bdeakyne
Just Black	3	1	DEVELOPMENT\VM-BEATS-DEV	a.archer
LastPass: Free Password Manager	4.79.0.3 4.80.0.3 4.81.0.2 4.82.0.2	1	DEVELOPMENT\VM-BEATS-DEV	a.archer
Legacy MindMup (discontinued)	0.0.0.20	1	SDE\USBO1SEFS-01	bdeakyne

Additional Resources

1. For more information on developer docs, see Carbon Black Cloud Splunk App
2. If you like to see an end-to-end demo on how to get Endpoint Events and Alerts into Splunk using the Carbon Black Cloud Event Forwarder, AWS S3+SQS, and the AWS Add-on for Splunk watch: Carbon Black Cloud & Splunk Integration

Change Log

The following updates were made to this guide:

Date	Description of Changes
1/07/21	

About the Author and Contributors

Bruce Deakyne is a Product Line Manager at VMware Carbon Black Cloud, focused on improving the ecosystem of APIs & integrations. Outside of cyber security, he enjoys cycling through the mountains of Boulder, CO.