```
WEBVTT

00:01.600 --> 00:05.720
In this video, we examine the component
architecture of QRadar SIEM.

00:06.960 --> 00:11.040
After looking at the high-level architecture,
we focus on the details of the Flow

00:11.080 --> 00:14.080
and Event Collectors, the Event Processor,
and the Console.

00:15.840 --> 00:18.320
Let's begin
by looking at the high-level architecture.

00:19.840 --> 00:24.920
Events from individual log sources and network
flow data are collected by the QRadar Event

00:24.960 --> 00:28.000
and Flow Collectors,
including QRadar Network Insights.

00:29.360 --> 00:33.200
Once the flow and event
data is forwarded to the Event or Flow Processor,

00:33.360 --> 00:36.360
it is stored in the Ariel database
on that same Processor.

00:36.720 --> 00:40.760
To fulfill the tamper-proof data storage aspects
for compliance mandates,

00:41.560 --> 00:46.360
data cannot be changed as soon as it is
stored in the Ariel flow or event database tables.

00:47.520 --> 00:52.880
If accumulation is required, the accumulated data
is stored in Ariel accumulation data tables.

00:53.880 --> 00:55.560
At any point in time,

00:55.600 --> 01:00.200
data can be selectively indexed
to support specific search and report requirements.

01:01.200 --> 01:07.160
```

```
If needed, one or multiple Data Nodes can be attached to the Event
Processor to extend the storage capacity.

01:08.760 --> 01:12.000
Once the event and Flow Processor has finished
processing the data,

01:12.240 --> 01:15.920
the QRadar Console services
continue with consolidated processing.

01:16.480 --> 01:19.560
Here, the Magistrate component creates and manages
offenses.

01:20.480 --> 01:24.600
These offenses, as well as asset, identity,
and configuration information,

01:24.840 --> 01:28.200
are stored in the Master PostgreSQL database
on the Console.

01:29.000 --> 01:35.440
In every QRadar SIEM deployment, there is one master
PostgreSQL database with optional read-only copies

01:35.480 --> 01:40.640
on each processor for backup and automatic restore
(these are not depicted in this diagram).

01:41.400 --> 01:45.960
Apps from the IBM Security App Exchange
can be hosted and run on the QRadar Console.

01:46.720 --> 01:50.040
In order to free up capacity
and improve performance on the Console,

01:50.240 --> 01:53.920
apps can be hosted on a dedicated App
Host appliance.

01:53.960 --> 01:59.120
Secure SSH communication between appliances
in a distributed environment is supported.

01:59.840 --> 02:02.160
Let's now look at the details of the Flow Collector.

02:03.640 --> 02:10.200
A network flow record provides information about a conversation
between two devices by using a specific protocol,
02:10.400 --> 02:13.680
and it can include fields
that provide details about the conversation.

02:14.680 --> 02:18.960
Examples include the source and destination
IP addresses, ports, and other fields.

02:19.960 --> 02:25.880
Flow data packets can be collected from a variety of network device
vendors and directly from the network interface.

02:26.920 --> 02:32.440
Collected flow data can update asset profiles
with the ports and services that are running on each host.

02:32.480 --> 02:40.040
If a new host is detected through network flow data,
a new asset is created in the QRadar asset database.

02:40.280 --> 02:42.480
Next in line is the Aggregator.

02:42.520 --> 02:48.480
This component enforces the license limit for the Flow Collector,
which is measured in Flows Per Minute (FPM).

02:49.440 --> 02:54.040
If the license limit is exceeded, flows are
temporarily stored in an overflow buffer,

02:54.040 --> 02:56.320
which processes them with the next set of flows.

02:56.960 --> 03:04.480
Additionally, an overflow record is created
with the source and destination address 127.0.0.4/5.

03:05.120 --> 03:08.760
Every flow source
protocol has an overflow buffer of 5 GB,

03:08.960 --> 03:12.520
and if the overflow buffer fills up,
the additional flows are dropped.

03:14.400 --> 03:18.640
More information about the QRadar licenses and handling the overflow
buffer

03:18.640 --> 03:22.040
was presented in the previous QRadar SIEM Architecture video.

03:23.160 --> 03:28.920
```

Then the Application Detection Module uses four methods to determine the application based on the flow data.

03:30.080 --> 03:32.480
The first is the User Defined method.

03:32.520 --> 03:37.160
This method is used when users have a proprietary
application that runs on their network.

03:37.720 --> 03:46.880
For example, all traffic that goes to host 10.100.100.42
on port 443 is recognized to be MySpecialApplication.

03:47.880 --> 03:50.600
The second method uses State-based decoders.

03:51.120 --> 03:53.920
This method is implemented
by looking at the source code.

03:54.600 --> 03:58.520
It determines the application
by analyzing the payload for multiple markers.

03:59.600 --> 04:08.480
For example, if you see A followed by B, then application = X;
and if you see A followed by C, then application = Y.

04:09.240 --> 04:11.240
The next method uses Signature matching.

04:11.760 --> 04:14.840
This method relies on basic string
matching in the payload.

04:15.400 --> 04:19.840
See the Applications configuration guide
for defining application mappings and signatures.

04:21.000 --> 04:23.480
The final method uses Port-based matching.

04:23.840 --> 04:29.960
In this case, applications are matched based
on their port use, for example, port 80 = HTTP.

04:31.280 --> 04:35.160
Finally, the flow data packets
reach the Flow reporting and routing component.

04:36.320 --> 04:39.400
This component is responsible
for creating superflows.

```
04:39.440 --> 04:44.240
Superflows only store a single flow record
with the collection of IP addresses,

04:44.280 --> 04:48.640
which allows processing of flows
to be faster and require less storage space.

04:49.600 --> 04:51.840
There are three types of superflows:

04:51.840 --> 04:58.200
Type A superflows contain a single source and multiple destination
addresses with the same destination port,

04:58.200 --> 05:00.920
byte count, and source flags or ICMP codes.

05:01.520 --> 05:04.400
An example for a type
A superflow is a network sweep.

05:04.920 --> 05:09.440
Type B superflows contain multiple sources
and a single destination address

05:09.680 --> 05:14.360
with the same destination port,
byte count, and source flags for ICMP codes.

05:15.480 --> 05:19.560
An example for a type B superflow is a
Distributed Denial of Service (DDoS) attack.

05:21.320 --> 05:23.840
And Type C superflows contain a single source

05:24.080 --> 05:27.560
and destination address
with changing source and destination ports.

05:28.200 --> 05:31.760
An example for a type C superflow is a port scan.

05:31.800 --> 05:37.520
Specific rule tests can leverage the flow type
to determine whether an offense needs to be created.

05:38.160 --> 05:40.280
The creation of superflows can be disabled.

05:41.480 --> 05:45.240
Up to a configurable number of bytes,
```

if the payload is unencrypted,

05:45.280 --> 05:49.040
QRadar QFlow provides
Layer 7 insights into the payload.

05:49.560 --> 05:55.880
By using a Tap or SPAN port, QFlow collects raw packets and places them into 60-second chunks.

05:56.920 --> 06:07.920
QFlow can also receive Layer 4 flows from other network devices in IPFIX/Netflow, sFlow, J-Flow Packeteer, and Flowlog file accounting technologies.

06:09.240 --> 06:11.120
Let's now look at the Event Collector.

06:13.040 --> 06:16.920
Each Event Collector gathers events from local and remote log sources.

06:17.720 --> 06:21.680
Once the raw data packets are received,
the license limit is checked first.

06:22.280 --> 06:26.640
On the Event Collector,
this limit is measured in events per second (EPS).

06:27.120 --> 06:33.880
If the EPS license is exceeded, they are temporarily stored in an overflow buffer and then processed during the next cycle.

06:34.400 --> 06:36.360
If the overflow buffer fills up,

06:36.400 --> 06:40.280
the additional events are dropped
and a message is logged for the administrators.

06:40.800 --> 06:44.600
More information about the QRadar licenses
and handling of the overflow

06:44.640 --> 06:48.400
buffer was presented in the previous QRadar
Architecture video.

06:49.680 --> 06:54.840
Log sources are automatically discovered after
record analysis in the Traffic Analysis module.

06:55.600 --> 07:00.200
This is an essential module

```
for automating a successful evaluation or deployment

07:00.400 --> 07:04.760
because it categorizes traffic from devices
that are unknown to the system.

07:05.600 --> 07:11.400
If detection is successful on an IP address, log
source detection creates a new QRadar log source.

07:12.640 --> 07:16.760
The Traffic Analysis module only carries out
detection on event protocols

07:16.800 --> 07:20.960
that are pushed to the Event Collector,
for example, syslog.

07:21.000 --> 07:25.360
After the correct log source is detected,
such as a Checkpoint firewall,

07:25.560 --> 07:28.440
the individual Device Support Modules parse
the events.

07:29.120 --> 07:32.120
First, the events are normalized
where source specific

07:32.160 --> 07:36.160
data fields are mapped into QRadar terminology
for further processing.

07:36.760 --> 07:44.200
The log source parser then extracts the log source event ID from the log
record and maps that to the QRadar identifier (QID).

07:44.880 --> 07:49.280
This is a unique ID that links the extracted
log source event ID to a QID.

07:50.400 --> 07:56.480
Each QID relates to a custom event name and description,
as well as severity and event category information.

07:56.920 --> 08:02.000
The event category information is structured into High Level Categories
(HLC) and Low Level Categories (LLC).

08:02.680 --> 08:07.760
For example, a valid category combination
is "Authentication" (being a High Level Category)
```

```
08:07.960 --> 08:11.120
and "Admin Login Successful"
(being a Low Level Category).

08:11.960 --> 08:17.800
Finally, the coalescing filter can optionally bundle identical events to
conserve system usage

08:17.800 --> 08:20.320
before handing off the data to the Event Processor.

08:22.760 --> 08:24.680
The Event Processor can receive

08:24.720 --> 08:29.520
event and flow data from Event and Flow Collectors
as well as other Event Processors

08:29.560 --> 08:33.120
that might be distributed
throughout the organization's IT deployment.

08:33.640 --> 08:38.560
First, the Overflow Filter enforces the license
in a similar way to the collectors.

08:39.320 --> 08:45.560
Next, the Custom Rule Engine (CRE) tests every event or flow against all
enabled rules.

08:45.960 --> 08:48.800
Matched rules might trigger the creation of an offense

08:49.280 --> 08:53.760
or create a CRE event
that then triggers the creation of an offense.

08:54.560 --> 08:59.960
Multiple matched events, flows, and matched rules
might correlate into a single offense,

09:00.000 --> 09:05.080
and a single event or flow
can be correlated into multiple offenses as well.

09:05.600 --> 09:12.880
By default, rules are tested against events or flows
that are received by a single Event Processor (local rules).

09:13.200 --> 09:16.960
However, Global Cross-Correlation (GCC)
allows rules testing across

09:17.000 --> 09:20.880
```

multiple Event Processors
in the QRadar SIEM deployment.

09:21.920 --> 09:26.320
From the CRE, the Exit Filter sends any events or flows
that have been marked

09:26.360 --> 09:29.960
for further processing
by the Magistrate component on the Console.

09:30.920 --> 09:34.520
Every event and flow is
then sent to the Event Storage Filter,

09:34.720 --> 09:38.160
where they are stored in the events or flows
Ariel database.

09:39.040 --> 09:46.960
If a new port or host is detected at this time, an asset profile needs to
be updated or created in the PostgreSQL database.

09:47.760 --> 09:53.040
The Host Profiler, also known as the Host
Profiling Filter, sends the collected information

09:53.080 --> 09:57.320
about the new host to the Console
so that an asset can be created or updated.

09:58.640 --> 10:01.960
Finally, if an analyst has defined any searches to collect

10:02.000 --> 10:06.480
and investigate specific sets of data, events
and flow records are accumulated

10:06.520 --> 10:10.400
every minute and stored in the accumulations
Ariel database tables.

10:10.960 --> 10:16.080
These accumulations are used to create time-series
statistical metadata for dashboards,

10:16.280 --> 10:22.200
event and flow forensics and searching, reporting,
and the Anomaly Detection Engine on the Console.

10:22.520 --> 10:27.240
Accumulated time intervals can be defined
as 1 minute, 1 hour, and 1 day.

```
10:28.520 --> 10:32.720
The Accumulator is a distributed component
that operates on each Event Processor.

10:35.480 --> 10:41.560
The Console receives data from the deployed Event Processors for further
analysis by the Magistrate component,

10:41.560 --> 10:45.960
which creates, manages, and stores offenses
in the PostgreSQL database.

10:47.320 --> 10:51.280
These offenses are then brought to the analyst's
attention in the user interface.

10:51.960 --> 10:55.600
The Magistrate instructs
the Ariel Proxy Server to gather information

10:55.640 --> 10:59.600
about all related events and flows
that triggered the creation of an offense.

11:00.320 --> 11:04.360
The collected data is then available
for further investigation by the analyst.

11:05.680 --> 11:11.160
If data is collected from multiple Event Processors,
the Console's Custom Rules Engine can use

11:11.200 --> 11:16.440
Global Cross-Correlation to test rules and data
from all deployed Event Processors.

11:17.880 --> 11:22.080
This helps to locate more complex attacks,
which can span across the overall

11:22.120 --> 11:26.960
IT infrastructure and are not confined
to being detected by a single Event Processor.

11:28.160 --> 11:32.240
The Vulnerability Information Server detects
and creates new assets

11:32.360 --> 11:36.280
or adds information,
such as open ports or discovered services

11:36.440 --> 11:41.840
to existing assets, based on information
```

from the Host Profiler on the Event Processors.

11:42.520 --> 11:48.800
This happens when hosts, services, or vulnerabilities that cannot be
mapped to existing assets are discovered.

11:49.360 --> 11:53.480
The Vulnerability Information Server
automatically checks the asset information

11:53.520 --> 11:57.920
against uploaded vulnerability information
by using flow information.

11:58.560 --> 12:05.640
The Anomaly Detection Engine searches the Accumulator databases for
anomalies which are then used for offense evaluation.

12:06.680 --> 12:09.800
There are three categories of Anomaly Detection
Rule types.

12:11.720 --> 12:18.000
The Threshold rule examines and numeric range,
such as greater than, less than, or a particular range.

12:18.800 --> 12:22.200
This rule can help detect the bandwidth of an application,

12:22.200 --> 12:27.840
the number of users connected to a VPN,
or a large and unusual outbound data transfer.

12:29.080 --> 12:34.560
The Anomaly rule looks at a change in short term
when comparing against a longer timeframe.

12:34.960 --> 12:40.400
This can help locate new service activity or
a change in the bandwidth volume on a specific link.

12:41.240 --> 12:47.080
And the Behavioral rule can detect changes from
the same time the day before or the previous week.

12:47.600 --> 12:54.560
This includes mail traffic. For example, the increase on external
or SMTP server traffic, which might be a relay.

12:55.440 --> 13:02.360
This rule can also be used for regular IT services, such as a backup
monitoring, where the rule would trigger if a backup failed.

13:04.240 --> 13:08.760

```
Let's now take a closer look at how offenses are being managed by the
Magistrate component.

13:10.200 --> 13:17.560
As seen earlier, rules can correlate events and flows into a single
offense, and a single event or flow can belong to multiple offenses.

13:18.800 --> 13:22.400
While rules are tested, they might lead to the creation
of an offense.

13:23.440 --> 13:26.040
Pending offenses tag the events and flows

13:26.040 --> 13:30.680
while the rule that triggered the creation of the offense remains at
least partially matched.

13:31.320 --> 13:34.440
A maximum of 100,000 offenses can be stored.

13:35.280 --> 13:41.240
Rules are stored in the PostgreSQL database
and replicated to Event and Flow Processors in your deployment.

13:42.400 --> 13:49.640
Events And flows that are tagged by the Custom Rules Engine for further
processing are handed over to the Console through the Exit Filter.

13:52.320 --> 13:56.600
Now that you know the details of every component
in a QRadar SIEM architecture,

13:56.720 --> 14:00.120
let's recap this section by examining a captured event

14:00.120 --> 14:02.960
from the time when it arrives at the Event Collector

14:02.960 --> 14:08.440
and follow it as it proceeds through correlation, accumulation and
storage on the Event Processor

14:08.440 --> 14:12.040
until it ends up as part of a larger offense on the Console.
```