LOGZILLA DOCUMENTATION

Advanced Event Correlation Walkthrough

Step-by-step walkthrough for building a LogZilla SEC Pair rule that tracks interface up/down events, calculates downtime, and alerts on extended outages

Event Correlation · Generated June 12, 2026 · logzilla.ai/docs/event-correlation/advanced-event-correlation-walkthrough

Overview

Event Correlation implementation using network interface monitoring provides administrators with practical experience creating correlation rules. The process involves tracking interface up/down events, calculating downtime durations, and generating critical alerts for extended outages.

Prerequisites: Before starting this walkthrough, review Event Correlation Overview (https://www.logzilla.ai/docs/event-correlation/event-correlation-overview) for setup requirements and enable Event Correlation on the LogZilla server.

Create a Test Correlation Rule

The following steps create a simple correlation rule that tracks network interface up/down events and calculates downtime.

Create the Correlation Rule File

Create the SEC instance directory structure and rule file:

```
sudo mkdir -p /etc/logzilla/sec/interface-monitoring/rules
sudo cat << 'EOF' > /etc/logzilla/sec/interface-monitoring/rules/interface-correlation.sec
# Interface Up/Down Correlation Rule
# Tracks interface state changes and calculates downtime
# Note: continue=TakeNext allows the same event to be processed by multiple rules

type=Pair
ptype=RegExp
continue=TakeNext
pattern=.*Interface (\S+) is down$
desc=Interface $1 Down - Starting Timer
action=eval %interface_name ( "$1" ); \
    eval %down_time ( time() ); \
    shellcmd (/usr/bin/logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT \
    -t EC-INTERFACE -p local0.info \
    "Interface $1 outage started at $(date)")
ptype2=RegExp
pattern2=.*Interface ($1) is up$
desc2=Interface $1 Up - Calculating Downtime
action2=eval %up_time ( time() ); \
    eval %downtime_seconds ( %up_time - %down_time ); \
    eval %downtime_minutes ( int( %downtime_seconds / 60 ) ); \
    shellcmd (/usr/bin/logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT \
    -t EC-INTERFACE -p local0.notice \
    "Interface %interface_name restored after %downtime_seconds seconds (%downtime_minutes minutes)")
EOF
```

Configure Event Forwarding to the correlation instance

Create a forwarder configuration to route interface events to the SEC instance:

Create Forwarder Configuration

Note: ensure the forwarding engine is enabled as noted in Event Correlation Overview (https://www.logzilla.ai/docs/event-correlation/event-correlation-overview)

```
sudo cat << 'EOF' > /etc/logzilla/forwarder.d/interface-monitoring.yaml
match:
  - field: message
    op: "=~"
    value: "Interface .* is .*"
type: sec
name: "Interface Monitoring Forwarder"
sec_name: interface-monitoring
EOF
```

Configuration Fields:

- `name`: Human-readable description for the forwarder (for debugging/identification)
- `sec_name`: Must match the SEC instance directory name (`/etc/logzilla/sec/interface-monitoring/`)

Verify Forwarder Rule

```
logzilla forwarder print
```

Apply Forwarder Changes

```
logzilla settings reload forwardermodule
```

Create a Critical Alert Rule

Add a second rule that generates critical alerts for extended outages:

Create Critical Alert Rule File

```
sudo cat << 'EOF' > /etc/logzilla/sec/interface-monitoring/rules/critical-alerts.sec
# Critical Alert for Extended Outages
# Alerts if interface stays down for more than 5 minutes

type=PairWithWindow
ptype=RegExp
pattern=.*Interface (\S+) is down$
desc=Interface $1 Down - Monitoring for Extended Outage
action=shellcmd (/usr/bin/logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT \
        -t EC-CRITICAL -p local0.alert \
        "CRITICAL: Interface $1 has been down for over 60 seconds")
ptype2=RegExp
pattern2=.*Interface ($1) is up$
desc2=Interface $1 Up - Canceling Critical Alert
action2=shellcmd (/usr/bin/logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT \
        -t EC-INTERFACE -p local0.info \
        "Interface $1 restored before critical threshold")
window=300
EOF
```

Create Test Event Generator

Create a script to generate test events that will trigger the correlation rules.

Create Test Script

```
sudo cat << 'EOF' > /tmp/test-interface-events.sh
#!/bin/bash

# Test Event Generator for Interface Correlation
# Generates interface up/down events to test correlation rules

INTERFACE_NAME="eth0"
LOGZILLA_HOST="127.0.0.1"  # Change to LogZilla server IP if running remotely
LOGZILLA_PORT="514"
LOGGER_CMD="/usr/bin/logger -T -n $LOGZILLA_HOST -P $LOGZILLA_PORT -t TestDevice -p local0.info"

echo "Starting interface correlation test..."
echo "This will generate interface down/up events with a 30-second gap"
echo "Watch LogZilla UI for correlation results"
echo ""

# Generate interface down event
echo "$(date): Sending interface down event"
$LOGGER_CMD "Interface $INTERFACE_NAME is down"
```

```
# Wait 30 seconds
echo "Waiting 30 seconds before sending up event..."
sleep 30

# Generate interface up event
echo "$(date): Sending interface up event"
$LOGGER_CMD "Interface $INTERFACE_NAME is up"

echo "Test complete. Check LogZilla for correlation results."
echo "Look for events with Program = EC-INTERFACE"
EOF
```

Make Script Executable

```
sudo chmod +x /tmp/test-interface-events.sh
```

Create Extended Outage Test

Create a second test for the critical alert rule:

Create Extended Test Script

```
sudo cat << 'EOF' > /tmp/test-critical-alert.sh
#!/bin/bash

# Extended Outage Test - Triggers Critical Alert
# Generates interface down event and waits 6 minutes

INTERFACE_NAME="eth1"
LOGZILLA_HOST="127.0.0.1"  # Change to LogZilla server IP if running remotely
LOGZILLA_PORT="514"
LOGGER_CMD="/usr/bin/logger -T -n $LOGZILLA_HOST -P $LOGZILLA_PORT -t TestDevice -p local0.info"

echo "Starting critical alert test..."
echo "This will generate an interface down event and wait 6 minutes"
echo "A critical alert should appear after 5 minutes"
echo ""

# Generate interface down event
echo "$(date): Sending interface down event for $INTERFACE_NAME"
$LOGGER_CMD "Interface $INTERFACE_NAME is down"

echo "Waiting 6 minutes to trigger critical alert..."
echo "Watch LogZilla UI - critical alert should appear after 5 minutes"
sleep 360
```

```
# Generate interface up event
echo "$(date): Sending interface up event"
$LOGGER_CMD "Interface $INTERFACE_NAME is up"

echo "Extended test complete."
EOF
```

Make Extended Script Executable

```
sudo chmod +x /tmp/test-critical-alert.sh
```

Restart Event Correlation Service

```
logzilla settings reload sec
```

Execute Tests and Verify Results

Run Basic Correlation Test

Execute the basic interface correlation test:

```
/tmp/test-interface-events.sh
```

Monitor Results in LogZilla UI

Open the LogZilla web interface
Navigate to the Events page
Apply one of these filters:

- Program = `TestDevice` (raw input events)
- Program = `EC-INTERFACE` (correlated output events)

Expected results:

- Initial down event with timestamp
- Correlation result showing downtime calculation

Run Critical Alert Test

In a separate terminal, execute the extended outage test:

```
/tmp/test-critical-alert.sh
```

Verify Critical Alert

In LogZilla UI, filter by Program = `EC-CRITICAL`
After 5 minutes, a critical alert should appear
The alert should show the interface name and duration

Understanding the Results

Expected Output

For the basic test, administrators should see events similar to:

```
Program: EC-INTERFACE
Message: Interface eth0 down at 1640995200

Program: EC-INTERFACE
Message: Interface eth0 restored after 30 seconds (0 minutes)
```

For the critical test:

```
Program: EC-CRITICAL
Message: CRITICAL: Interface eth1 has been down for 5 minutes
```

How the Correlation Works

Pair Rule: Matches interface down events and waits for corresponding up events
Time Calculation: Event Correlation calculates the difference between down and up timestamps
Event Generation: New enriched events are sent back to LogZilla with calculated downtime
Window Rule: Monitors for events that don't have a matching pair within the specified time window

Advanced Configuration

Modify Time Windows

To change the critical alert threshold, edit the rule file:

```
sudo nano /etc/logzilla/sec/interface-monitoring/rules/critical-alerts.sec
```

Change `window=300` to desired seconds (e.g., `window=600` for 10 minutes).

Add More Complex Patterns

Administrators can extend rules to match different event formats:

```
# Match multiple interface formats
pattern=(Interface|Port|Link) (\S+) (is down|down|failed)
```

Apply Configuration Changes

After making changes to correlation rules, restart the Event Correlation service to load the new configuration:

```
sudo logzilla restart -c sec
```

Troubleshooting

Verify Basic System Health

First, ensure the SEC container is running and accessible:

```
# Check SEC container status
sudo docker ps | grep lz_sec

# Review SEC logs for errors
sudo docker logs lz_sec
```

Validate Configuration

Verify Forwarder Configuration

Verify the forwarder configuration is syntactically correct:

```
logzilla forwarder print
```

This command displays the parsed configuration and reports any syntax errors.

Verify SEC Rule Syntax

Validate the SEC rule files for syntax errors:

```
sudo docker exec -it lz_sec sec --testonly --conf=/etc/logzilla/sec/interface-
monitoring/rules/interface-correlation.sec
```

This command parses the SEC rule file and reports any syntax errors without starting the correlation engine.

Test Event Flow

1. Verify Events Reach LogZilla

Check that the test events are being received:

- Navigate to the Events page in LogZilla UI
- Filter by Program = `TestDevice`
- Confirm interface up/down events appear

2. Test SEC Connectivity

Verify SEC can send events back to LogZilla:

```
# Check SEC environment variables
sudo docker exec -it lz_sec printenv | grep SYSLOG

# Test TCP connectivity (preferred)
sudo docker exec -it lz_sec sh -c \
  '/usr/bin/logger -T -n "$SYSLOG_HOSTNAME" -P "$SYSLOG_BSD_TCP_PORT" \
  -t EC-TEST "SEC connectivity test"'
```

If the test event doesn't appear in LogZilla, try UDP as a fallback:

```
sudo docker exec -it lz_sec sh -c \
  '/usr/bin/logger -n "$SYSLOG_HOSTNAME" -P "$SYSLOG_BSD_UDP_PORT" \
  -t EC-TEST "SEC UDP connectivity test"'
```

Note: Using `sh -c` ensures environment variables are expanded inside the container.

Common Issues

Events Not Correlating

- **Pattern mismatch**: Ensure the SEC rule patterns match the exact event format. Test events may include additional fields (hostname, program) that need to be accounted for in patterns.
- **Timing issues**: SEC rules have time windows - events outside the window won't correlate
- **Case sensitivity**: SEC pattern matching is case-sensitive
- **Events not reaching SEC**: Check forwarder configuration and ensure events match the forwarder's filter criteria

No Correlation Output

- **SEC rules not loading**: Check `docker logs lz_sec` for rule syntax errors
- **Forwarder not routing**: Verify forwarder configuration with `logzilla forwarder print`
- **Network connectivity**: Test SEC → LogZilla connectivity using the test commands above

Pattern Debugging Example

If SEC receives events like:

```
clitests TestDevice Interface eth0 is down
```

But the rule pattern is:

```
pattern=Interface (\S+) is down$
```

The pattern won't match. Fix it with:

```
pattern=.*Interface (\S+) is down$
```

Related Documentation

- Event Correlation Rule Types (https://www.logzilla.ai/docs/event-correlation/event-correlation-rule-types)
- Creating Triggers (https://www.logzilla.ai/docs/creating-triggers/)