LOGZILLA DOCUMENTATION

Forwarding to Event Correlation

Route selected LogZilla events to SEC instances using type sec forwarders and sec_name, keeping stateful correlation focused on relevant traffic

Forwarding To Downstream Receivers · Generated June 12, 2026 · logzilla.ai/docs/forwarding-module/forwarding-to-event-correlation

Forwarding to Event Correlation (SEC)

Purpose

Forwarders control which events LogZilla sends to Event Correlation (SEC) for stateful processing. The sections below cover how to use `type: sec` forwarders, how `sec_name` maps to SEC instances under `/etc/logzilla/sec/`, and how to verify that events reach SEC.

Prerequisites

- Event Correlation enabled in LogZilla settings
  (see Event Correlation Overview (https://www.logzilla.ai/docs/event-correlation/event-correlation-overview)).
- Forwarder module enabled:

```
sudo logzilla settings update FORWARDER_ENABLED=true
```

- Basic familiarity with forwarder configuration
  (see Forwarder and Deduplication Overview (https://www.logzilla.ai/docs/forwarding-module/dedup-forwarder-introduction)).

How SEC Forwarders Work

LogZilla uses selective event forwarding to optimize SEC performance:

Event ingestion: LogZilla receives and parses incoming events.
Forwarder evaluation: Events are evaluated against forwarder rules.
Selective routing: Only matching events are sent to SEC instances.
SEC processing: SEC rules correlate the forwarded events.
Response injection: SEC results are sent back into LogZilla as new events.
This approach ensures that SEC focuses only on relevant events and avoids unnecessary load from unrelated traffic.

Forwarder Type `sec`

The `sec` forwarder type sends events from LogZilla to a specific SEC instance.

Basic SEC Forwarder Example

Forward all events from a specific program to SEC:

```
# /etc/logzilla/forwarder.d/test-events.yaml
match:
  - field: program
    op: "eq"
    value: "TEST_APP"
type: sec
name: "Test Application Forwarder"
sec_name: example
```

Configuration breakdown:

- `match`: Defines filtering criteria for events.
- `field: program`: Matches the syslog program field.
- `op: "eq"`: Uses the equality operator.
- `value: "TEST_APP"`: Matches events where `program` equals `TEST_APP`.
- `type: sec`: Sends events to a SEC instance instead of an external receiver.
- `name`: Human-readable description for troubleshooting.
- `sec_name: example`: Must match a SEC instance directory name under `/etc/logzilla/sec/`.

Message Rewriting Before SEC

Transform events before sending them to SEC to simplify correlation rules:

```
# /etc/logzilla/forwarder.d/auth-events.yaml
match:
  - field: program
    op: "eq"
    value: ["sshd", "sudo", "su"]
type: sec
name: "Authentication Events"
sec_name: security-monitoring
rules:
  - match:
      - field: message
        op: "=~"
        value: "Failed password"
    rewrite:
      message: "AUTH_FAILED $MESSAGE"
  - match:
      - field: message
        op: "=~"
        value: "Accepted password"
    rewrite:
      message: "AUTH_SUCCESS $MESSAGE"
```

This prepends `AUTH_FAILED` or `AUTH_SUCCESS` to messages, allowing SEC rules to use simple substring matching (`pattern=AUTH_FAILED`) instead of complex regular expressions.

Multiple SEC Instances

Route different event types to specialized SEC instances:

```
# /etc/logzilla/forwarder.d/network-forwarding.yaml
match:
  - field: program
    op: "eq"
    value: "cisco_ios"
  - field: cisco_mnemonic
    op: "eq"
    value: "BGP-5-ADJCHANGE"
type: sec
name: "BGP Events"
sec_name: bgp-monitoring

---
# Same file, different forwarder
match:
  - field: program
    op: "eq"
    value: "cisco_ios"
  - field: cisco_mnemonic
    op: "=~"
    value: "LINK-.*-UPDOWN"
type: sec
name: "Interface Events"
sec_name: interface-monitoring
```

This separates BGP correlation from interface correlation, allowing independent rule management and scaling.

Forwarder and SEC Instance Relationship

The `sec_name` value in a SEC forwarder configuration must exactly match a directory name under `/etc/logzilla/sec/`.

Example directory structure:

```
/etc/logzilla/sec/
├── example/
│   └── rules/
│       └── test.sec
├── network-monitoring/
│   └── rules/
│       ├── bgp-correlation.sec
```

```
│       └── interface-correlation.sec
└── windows-security/
    └── rules/
        └── brute-force.sec
```

Corresponding forwarder `sec_name` values:

- `sec_name: example` → `/etc/logzilla/sec/example/`
- `sec_name: network-monitoring` → `/etc/logzilla/sec/network-monitoring/`
- `sec_name: windows-security` → `/etc/logzilla/sec/windows-security/`

Applying SEC Forwarder Changes

After creating or modifying SEC forwarder configurations:

```
# Validate configuration syntax
logzilla forwarder print

# Apply changes via settings reload
sudo logzilla settings reload forwardermodule

# Verify forwarder is processing
sudo docker logs lz_forwarder | tail -20
```

The `logzilla forwarder print` command displays the merged forwarder configuration and reports syntax errors before applying changes.

Verifying Forwarding to SEC

To confirm that events are being forwarded to SEC:

```
# Monitor SEC logs for incoming events
sudo docker logs lz_sec -f

# Send test event from a client to LogZilla
logger -n <logzilla-ip> -P 514 -t TEST_APP "TEST_EVENT: Sample message"

# SEC logs should show HTTP requests from the forwarder
# Example:
# 2025-10-20 11:15:23 [1] uvicorn.access INFO 172.17.0.1:12345 \
#   "POST /sec/example HTTP/1.1" 200
```

If SEC logs show no activity after sending test events:

Verify the forwarder is enabled:

```
sudo logzilla settings list | grep FORWARDER_ENABLED
```

Check forwarder configuration:

```
logzilla forwarder print
```

Verify the SEC instance exists:

```
ls -la /etc/logzilla/sec/
```

Reload both services:

```
sudo logzilla settings reload sec forwardermodule
```

Complete Working Example

This example demonstrates a complete SEC instance and SEC forwarder for testing.

1. Create SEC Instance and Rule

```
# Create SEC instance directory structure
sudo mkdir -p /etc/logzilla/sec/example/rules

# Create simple SEC rule
sudo cat << 'EOF' > /etc/logzilla/sec/example/rules/test.sec
# Count TEST events and alert at threshold
type=SingleWithThreshold
ptype=SubStr
pattern=TEST_EVENT
desc=SEC Test Rule Triggered
action=shellcmd (logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT \
       -t SEC-ALERT -p local0.alert \
       "SEC_WORKING: Detected $thresh test events in $window seconds")
thresh=3
window=60
EOF
```

2. Create SEC Forwarder Configuration

```
sudo cat << 'EOF' > /etc/logzilla/forwarder.d/test.yaml
match:
  - field: program
    op: "eq"
    value: "TEST_APP"
type: sec
name: "Test SEC Forwarder"
sec_name: example
EOF
```

3. Validate and Apply

```
# Validate configuration
logzilla forwarder print

# Apply changes
sudo logzilla settings reload sec forwardermodule

# Verify SEC container is running
sudo docker ps | grep lz_sec
```

4. Test Correlation

```
# Monitor SEC logs
sudo docker logs lz_sec -f &

# Send three test events
for i in {1..3}; do
  logger -n <logzilla-ip> -P 514 -t TEST_APP "TEST_EVENT: Test message $i"
  sleep 2
done

# Check LogZilla UI for SEC-ALERT message
# Or search logs: sudo docker logs lz_sec | grep "SEC_WORKING"
```