LOGZILLA DOCUMENTATION

Event Correlation Overview

Enable LogZilla Event Correlation to detect multi-event patterns across time using stateless triggers and stateful SEC rules with forwarder routing

Event Correlation · Generated April 29, 2026 · logzilla.ai/docs/event-correlation/event-correlation-overview

Event Correlation Overview

Event correlation identifies patterns, relationships, and anomalies across multiple log events over time. LogZilla provides both stateless and stateful correlation methods to detect complex scenarios that single-event analysis cannot identify.

Before you begin

- Enable EC (required):

```
sudo logzilla settings update SEC_ENABLED=true
```

- Enable Forwarder (required):

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

- File locations:
  - EC rules: `/etc/logzilla/sec/<instance>/rules/`
  - Forwarder configs: `/etc/logzilla/forwarder.d/`
- Apply changes:
  - Forwarder configs: `logzilla settings reload forwardermodule`
  - EC rule changes: `logzilla settings reload sec`
  - Both together: `logzilla settings reload sec forwardermodule`
- Validate forwarder configuration:

```
logzilla forwarder print
```

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

- Verify EC container:

```
sudo docker ps | grep lz_sec
```

Container Logging

- EC rules use environment variables `$SYSLOG_HOSTNAME` and `$SYSLOG_BSD_TCP_PORT` to target the LogZilla syslog receiver
- The `-T` flag enables TCP transport; omit for UDP if TCP is not supported
- EC runs in a container with limited utilities; use inline Perl for complex operations rather than external commands
- DNS resolution occurs within the EC container; ensure proper DNS configuration (https://www.logzilla.ai/docs/administration/custom-dns) for hostname lookups

Core Correlation Concepts

- **Event triggers**: Conditions that initiate correlation analysis
- **Event filters**: Criteria that determine which events to correlate
- **Event pairing**: Associations between multiple related events
- **Time windows**: Duration constraints for event relationships
- **Thresholds**: Frequency or count-based correlation triggers
- **Suppression**: Logic to reduce noise and prevent alert storms

LogZilla Correlation Architecture

LogZilla implements a two-tier correlation system:

Stateless Correlation (Triggers)

LogZilla Triggers provide immediate, single-event correlation:

- Events are matched against trigger filters in real-time
- Actions execute immediately when conditions are met
- Suitable for alerting and immediate responses
- Managed through the LogZilla web interface or API

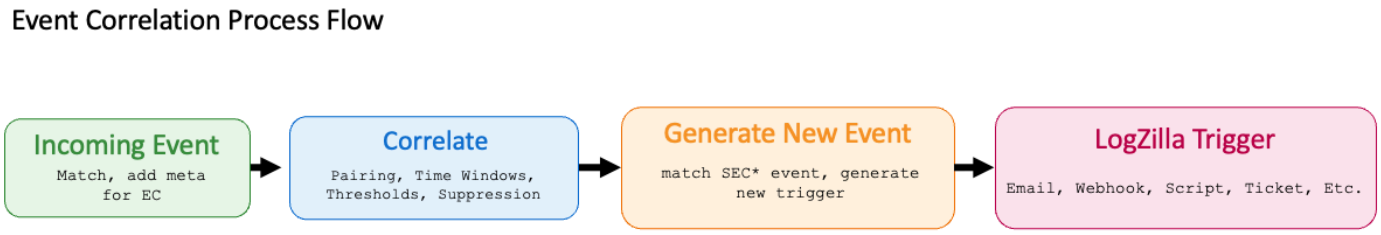
Stateful Correlation

Event Correlation enables advanced, multi-event correlation:

- Maintains state across multiple events over time
- Supports complex patterns like event pairs, thresholds, and sequences
- Handles scenarios requiring memory of past events
- Integrated via the Script Server for scalable processing

Event Processing Flow

Event Ingestion: Incoming events (syslog, snmp, httpx, webhooks, etc.) are received and forwarded to ParserModule
Parsing: Events are normalized and enriched with user tags via Parser Rules (YAML), Lua scripts, or Apps.
Trigger Evaluation: QueryModule matches events against active triggers
Action Execution: Matched triggers execute configured actions
Event Correlation Processing: Complex correlation rules process events via Script Server
Response: Correlated events inject a new event back into LogZilla for output triggering (Email, Webhook, Scripts, etc.)



Scalable Correlation Design

The architecture addresses traditional EC limitations:

- **Selective forwarding**: Only relevant events are sent to EC for processing
- **Distributed processing**: Script Server manages multiple EC instances
- **Event filtering**: Triggers pre-filter events before correlation analysis
- **Resource optimization**: Correlation load is distributed and controlled

Use Cases

Network Infrastructure

- **Interface flapping**: Detect rapid up/down state changes
- **Device failures**: Correlate multiple error conditions
- **Security incidents**: Identify attack patterns across devices

System Monitoring

- **Service failures**: Detect cascading service dependencies
- **Performance degradation**: Correlate resource exhaustion indicators
- **Unauthorized access**: Track suspicious login patterns

Application Monitoring

- **Error clustering**: Group related application errors
- **Transaction failures**: Correlate multi-step process failures
- **Capacity planning**: Identify usage pattern trends

Getting Started

Event correlation in LogZilla involves:

Configure triggers: Create stateless correlation rules for immediate responses
Define EC rules: Develop stateful correlation patterns for complex scenarios
Test correlation: Verify rules work with sample events
Monitor performance: Ensure correlation processing scales appropriately

LogZilla Pre-Processing Performance Advantages

LogZilla's architecture provides significant performance benefits for event correlation:

Traditional EC Approach Limitations

- **Complex regex parsing**: EC must parse raw log messages with CPU-intensive regex
- **Duplicate parsing logic**: Same parsing repeated across multiple EC rules
- **Limited context**: Only raw message text available for correlation
- **Performance bottlenecks**: EC spends most CPU time on text parsing, not correlation
- **Alert fatigue**: Thousands of individual event alerts daily
- **Missed attack patterns**: Single events don't show attack progression
- **Manual correlation**: Analysts must manually connect related events

LogZilla + EC Architecture Benefits

- **Pre-extracted fields**: LogZilla's Lua engine extracts fields once, EC accesses via environment variables
- **5x-10x performance improvement**: EC focuses on correlation logic instead of parsing
- **Rich context**: EC receives structured data like `EVENT_USER_TAGS_PAM_USER_TRACKING`, `EVENT_USER_TAGS_SRCIP`
- **Centralized parsing**: One optimized parsing engine serves all LogZilla features
- **Simplified rules**: EC rules use simple environment variable checks instead of complex regex
- **Attack pattern detection**: Multi-stage attack identification

- **Intelligent response**: Context-aware automation based on external data
- **Reduced noise**: Correlation replaces individual event alerts
- **Separation of concerns**: EC focuses on correlation, triggers handle actions

Example Performance Comparison

Traditional EC (inefficient):

```
pattern=.*sshd.*Failed password for (\w+) from ([\d.]+) port (\d+)
```

LogZilla Pre-Processing (efficient):

```
pattern=AUTH_FAILED
# Access: $ENV{EVENT_USER_TAGS_PAM_USER_TRACKING}, $ENV{EVENT_USER_TAGS_SRCIP}
```

Script Execution Context

EC vs LogZilla Trigger Scripts

Distinction: Scripts called by EC and LogZilla Triggers receive data in different ways:

EC `shellcmd` Scripts

- **Data Access**: Command-line arguments (`$1`, `$2`, `$3`, etc.)
- **Usage**: `shellcmd /path/to/script.sh "%variable1" "%variable2"`
- **Script**: `USERNAME="$1"; SRC_IP="$2"`

LogZilla Trigger Scripts

- **Data Access**: Environment variables (`$EVENT_*`)
- **Usage**: `exec_script: true` in trigger configuration
- **Script**: `USERNAME="$EVENT_USER_TAGS_PAM_USER_TRACKING"`

Important: Do not mix these execution contexts. EC scripts cannot access `$EVENT_*` environment variables.

Quick reference

- You must use a LogZilla forwarder configuration to send events to SEC.
- The forwarder can use a rewrite rule to alter the message contents to be sent to SEC, for example to send user tags.
- The forwarder should use an appropriate `match` configuration to limit the rate at which events are sent to SEC, to only what is necessary.

- SEC `shellcmd` scripts: receive values as positional arguments (`$1`, `$2`, ...). Pass required variables explicitly from the rule, for example: `shellcmd (/path/to/script.sh "%username" "%src_ip")`.

Prerequisites

Event Correlation implementation requires:

- LogZilla server with administrative access
- EC feature enabled in LogZilla settings
- Basic understanding of regular expressions
- SSH access to the LogZilla server

Enabling EC

Check EC Status

Verify if EC is enabled on the LogZilla server:

```
sudo logzilla settings list | grep SEC_ENABLED
```

Enable EC

If EC is not enabled, activate it:

```
sudo logzilla settings update SEC_ENABLED=true
```

Once EC is enabled, the container starts automatically. This process takes approximately one minute.

Enable Event Forwarding

Event Correlation relies on the Forwarder module to send selected events to SEC instances.

Enable the forwarder module:

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

For detailed forwarder configuration and examples, see:

- Forwarder and Deduplication Overview (https://www.logzilla.ai/docs/forwarding-module/dedup-forwarder-introduction)
- Downstream Syslog Receivers (https://www.logzilla.ai/docs/forwarding-module/downstream-syslog-receivers)
- Forwarding to Event Correlation (https://www.logzilla.ai/docs/forwarding-module/forwarding-to-event-correlation)

Verify EC Container

Verify the container has started:

```
sudo docker ps | grep lz_sec
```

The output should show the EC container is running:

```
lz_sec    logzilla/script-server:v6.37.4   "/usr/bin/run_script..."   About a minute ago   Up About a
minute
```

EC Configuration Management

Rule Loading

EC automatically loads new rules from `/etc/logzilla/sec/` when started. Verify EC is running:

```
sudo docker ps | grep lz_sec
```

Verify Rule Processing

Check EC logs to confirm rules are loaded:

```
sudo docker logs lz_sec | tail -20
```

Look for messages indicating successful rule loading.

Apply Configuration Changes

After making changes to EC rules, reload the SEC configuration:

```
sudo logzilla settings reload sec
```

If a reload does not pick up changes (rare), then restart the SEC container:

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

Troubleshooting

EC Not Processing Events

Check EC container status:

```
sudo docker ps | grep lz_sec
```

Review EC logs:

```
sudo docker logs lz_sec
```

Verify rule syntax:

```
sudo docker exec -it lz_sec sec --testonly --conf=/etc/logzilla/sec/rule-file.sec
```

Events Not Correlating

Verify test events are reaching LogZilla:

- Check Events page for original test events
- Ensure expected program names appear

Check pattern matching:

- Ensure event format matches EC rule patterns
- Use exact string matching when possible for better performance

No Correlation Output

Verify EC can send events back to LogZilla:

```
sudo docker exec -it lz_sec printenv | grep SYSLOG
```

Test logger connectivity from EC container:

```
sudo docker exec -it lz_sec /usr/bin/logger -T -n $SYSLOG_HOSTNAME -P $SYSLOG_BSD_TCP_PORT -t EC-TEST
"Test message"
```

Technical Notes

- EC rules use environment variables `$SYSLOG_HOSTNAME` and `$SYSLOG_BSD_TCP_PORT` to target the LogZilla syslog receiver
- The `-T` flag enables TCP transport; omit for UDP if TCP is not supported
- EC runs in a container with limited utilities; use inline Perl for complex operations rather than external commands
- DNS resolution occurs within the EC container; ensure proper DNS configuration for hostname lookups

Related Topics

- Event Correlation Rule Types (https://www.logzilla.ai/docs/event-correlation/event-correlation-rule-types)
- Windows Event Correlation (https://www.logzilla.ai/docs/event-correlation/windows-event-correlation)
- Advanced Event Correlation Walkthrough (https://www.logzilla.ai/docs/event-correlation/advanced-event-correlation-walkthrough)
- Creating Triggers (https://www.logzilla.ai/docs/creating-triggers/)