LOGZILLA DOCUMENTATION

# Brute Force Attack Detection

Detect SSH, FTP, RDP, and web brute-force attacks in LogZilla by correlating failed authentication attempts across PAM, sshd, and web server logs

Event Correlation · Generated April 29, 2026 · logzilla.ai/docs/event-correlation/brute-force-attack-detection

# Brute Force Attack Detection and Response

Brute force attack detection requires correlating multiple failed authentication attempts over time, followed by automated responses. The architecture separates correlation logic (SEC) from response automation (triggers), enabling structured response workflows.

Prerequisites: Ensure Event Correlation is enabled and forwarder reloading is available as shown in the Event Correlation Overview (https://www.logzilla.ai/docs/event-correlation/event-correlation-overview).

# Basic Brute Force Detection

## LogZilla Forwarder Configuration

**Required App**: `linux` app for basic authentication failure detection and `linux__pam` app (for `PAM User Tracking` and `PAM Remote Host` user tags).

```
# /etc/logzilla/forwarder.d/linux-auth.yaml
window_size: 0
type: sec
sec_name: linux-auth
match:
  - field: program
    value:
      - "sshd"
      - "vsftpd"
      - "proftpd"
      - "pure-ftpd"
      - "httpd"
      - "apache2"
      - "nginx"
      - "xrdp"
rules:
- rewrite:
    message: >-
      [LINUX_AUTH]
      program="${:program}"
      host="${:host}"
      srcip="${:SrcIP}"
      username="${:Username}"
      auth_success="${:Auth Success}"
      $MESSAGE
```

## SEC Rule: Failed Login Threshold Detection

```
# /etc/logzilla/sec/linux-auth/failed-login-detection.sec
# ============================================
# Failed Login Detection
# Detects multiple failed login attempts indicating brute force attack
# ============================================

type=SingleWithThreshold
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="false"
desc=Failed login for $4 from $3
thresh=5
window=1800
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
       create FAILED_LOGIN_ACTIVE_%username_%srcip 1800; \
       write /var/log/logzilla/sec/security.log "BRUTE_FORCE_DETECTED %t program=%program host=%host
srcip=%srcip username=%username"; \
       shellcmd logger -t SEC-SECURITY -p local0.warning "BRUTE_FORCE_DETECTED program=\"%program\"
host=\"%host\" srcip=\"%srcip\" username=\"%username\" - possible brute force attack"
```

## LogZilla Trigger: Brute Force Response

```
name: "Brute Force Attack Response"
filter:
  - field: program
    op: eq
    value: SEC-SECURITY
  - field: message
    op: "=~"
    value: "BRUTE_FORCE_DETECTED"
actions:
  exec_script: true
  script_path: "/usr/local/bin/brute-force-response.sh"
  issue_notification: true
  send_webhook: true
  send_webhook_template: |
    {
      "alert_type": "brute_force_attack",
      "username": "{{:Username}}",
      "source_ip": "{{:SrcIP}}",
      "target_host": "{{:host}}",
      "service": "{{:program}}",
      "attempts": "5"
    }
```

**Note**: The trigger filter now correctly matches `BRUTE_FORCE_DETECTED` from the SEC rule output.

## Intelligent Response Script

```
#!/bin/bash
# /usr/local/bin/brute-force-response.sh
# Called by LogZilla trigger - receives data via environment variables

# Parse user tags from EVENT_USER_TAGS (format: key1=value1,key2=value2)
parse_user_tags() {
    local tags="$1"
    local IFS=','
    for pair in $tags; do
        local key="${pair%%=*}"
        local value="${pair#*=}"
        case "$key" in
            Username) USERNAME="$value" ;;
            SrcIP) SRC_IP="$value" ;;
        esac
    done
}

# Validate required environment variables
if [[ -z "$EVENT_USER_TAGS" ]] || [[ -z "$EVENT_MESSAGE" ]]; then
    logger -t SECURITY-RESPONSE "ERROR: Missing required environment variables"
    exit 1
fi

# Extract data from environment variables
parse_user_tags "$EVENT_USER_TAGS"
TARGET_HOST="$EVENT_HOST"

# Parse attempt count and program from SEC message
# Format: BRUTE_FORCE_DETECTED ... program="sshd" host="server1" srcip="1.2.3.4" username="admin"
attempts="5" window="300s"
PROGRAM=$(echo "$EVENT_MESSAGE" | grep -oP 'program="\K[^"]+')
ATTEMPT_COUNT=$(echo "$EVENT_MESSAGE" | grep -oP 'attempts="\K[^"]+')

# Validate we got required data
if [[ -z "$USERNAME" ]] || [[ -z "$SRC_IP" ]]; then
    logger -t SECURITY-RESPONSE "ERROR: Could not parse Username or SrcIP from user tags:
$EVENT_USER_TAGS"
    exit 1
fi

# Query threat intelligence for IP reputation
IP_REPUTATION=$(curl -s "https://threat-intel.company.com/ip/$SRC_IP" 2>/dev/null || echo "unknown")

# Check if user account is service account or human
ACCOUNT_TYPE=$(curl -s "https://identity.company.com/api/account-type/$USERNAME" 2>/dev/null || echo
"unknown")
```

```
# Make intelligent blocking decision
if [[ "$IP_REPUTATION" == "known_malicious" ]]; then
    # Block immediately for known bad IPs
    iptables -I INPUT -s "$SRC_IP" -j DROP 2>/dev/null
    logger -t SECURITY-RESPONSE "Blocked known malicious IP: $SRC_IP attacking $USERNAME@$TARGET_HOST
via $PROGRAM"

    # Create high-priority incident
    curl -s -X POST "https://servicedesk.company.com/api/incidents" \
         -H "Content-Type: application/json" \
         -d "{\"priority\":\"high\",\"subject\":\"Malicious IP Brute Force\",\"details\":\"IP:
$SRC_IP, Target: $USERNAME@$TARGET_HOST, Service: $PROGRAM, Attempts: $ATTEMPT_COUNT\"}" \
         2>/dev/null

elif [[ "$ACCOUNT_TYPE" == "service_account" ]]; then
    # Service account attacks are high priority
    logger -t SECURITY-RESPONSE "Service account brute force: $USERNAME from $SRC_IP ($ATTEMPT_COUNT
attempts via $PROGRAM)"

    # Temporarily disable service account
    curl -s -X POST "https://identity.company.com/api/disable-account" \
         -H "Content-Type: application/json" \
         -d "{\"username\":\"$USERNAME\",\"reason\":\"brute_force\",\"source_ip\":\"$SRC_IP\"}" \
         2>/dev/null

    # Alert operations team immediately
    curl -s -X POST "https://slack.company.com/api/webhooks/ops-alerts" \
         -H "Content-Type: application/json" \
         -d "{\"text\":\"🚨 URGENT: Service account $USERNAME under brute force attack from $SRC_IP
($ATTEMPT_COUNT attempts via $PROGRAM on $TARGET_HOST)\"}" \
         2>/dev/null

else
    # Standard user account - rate limit and monitor
    # Implement temporary IP blocking with auto-expiry
    iptables -I INPUT -s "$SRC_IP" -j DROP 2>/dev/null
    echo "iptables -D INPUT -s $SRC_IP -j DROP 2>/dev/null" | at now + 1 hour 2>/dev/null

    logger -t SECURITY-RESPONSE "Temporarily blocked $SRC_IP for brute force against
$USERNAME@$TARGET_HOST ($ATTEMPT_COUNT attempts via $PROGRAM)"

    # Create standard priority ticket
    curl -s -X POST "https://servicedesk.company.com/api/tickets" \
         -H "Content-Type: application/json" \
         -d "{\"priority\":\"medium\",\"subject\":\"Brute Force Attack\",\"details\":\"User:
$USERNAME, IP: $SRC_IP, Host: $TARGET_HOST, Service: $PROGRAM, Attempts: $ATTEMPT_COUNT\"}" \
         2>/dev/null
fi

# Log to SEC tracking file for auditing
echo "$(date -
Iseconds)|$USERNAME|$SRC_IP|$TARGET_HOST|$PROGRAM|$ATTEMPT_COUNT|$IP_REPUTATION|$ACCOUNT_TYPE" >>
/var/log/logzilla/sec/auth-failure-responses.log
```

```
exit 0
```

# Advanced Brute Force Patterns

## Distributed Brute Force Detection

Detect coordinated attacks from multiple source IPs targeting the same user.

## SEC Rule: Distributed Attack Detection

```
# /etc/logzilla/sec/linux-auth/distributed-brute-force-detection.sec
# ==========================================
# Distributed Brute Force Detection
# Tracks failed login attempts per username from multiple source IPs
# Detects coordinated attacks - alerts only, no blocking
# ==========================================

type=SingleWithThreshold
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="false"
desc=Distributed brute force attack detected against user $4
thresh=20
window=600
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
       write /var/log/logzilla/sec/security.log "DISTRIBUTED_BRUTE_FORCE %t program=%program
host=%host username=%username latest_srcip=%srcip total_attempts=%thresh window=600s"; \
       shellcmd logger -t SEC-SECURITY -p local0.warning "DISTRIBUTED_BRUTE_FORCE
username=\"%username\" host=\"%host\" latest_srcip=\"%srcip\" total_attempts=\"%thresh\"
window=\"600s\" - distributed attack from multiple IPs detected"
```

## Password Spray Detection

Detect password spray attacks (same password against multiple users).

## SEC Rule: Password Spray Detection

```
# /etc/logzilla/sec/linux-auth/password-spray-detection.sec
# Password Spray Attack Detection
# Detects when single IP attempts authentication as multiple different users
```

```
# Indicates password spray attack

type=SingleWithThreshold
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="false"
desc=Password spray attack detected from $3
thresh=50
window=1800
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
       write /var/log/logzilla/sec/security.log "PASSWORD_SPRAY_DETECTED %t srcip=%srcip
total_attempts=50 window=1800s - single IP attacking multiple accounts"; \
       shellcmd logger -t SEC-SECURITY -p local0.alert "PASSWORD_SPRAY_DETECTED srcip=\"%srcip\"
username=\"%username\" host=\"%host\" program=\"%program\" total_attempts=50 window=1800s - single IP
attacking multiple accounts"
```

# Successful Login After Brute Force

## Compromise Detection Pattern

Detect successful logins that occur shortly after brute force attempts, indicating potential credential compromise.

## SEC Rule: Post-Brute Force Success Detection

You must also use the SEC rule indicated in *Failed Login Threshold Detection* above, because it tracks the repeated failed logins before success.

```
# /etc/logzilla/sec/linux-auth/brute-force-success.sec
# ============================================
# Successful Login After Failed Attempts
# Detects when user successfully logs in after multiple failures
# Indicates potential successful brute force / credential compromise
# ============================================

type=Single
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="true"
desc=Successful login for $4 from $3 after previous brute force attempts
context=FAILED_LOGIN_ACTIVE_$4_$3
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
```

```
        write /var/log/logzilla/sec/security.log "CREDENTIAL_COMPROMISE_SUSPECTED %t program=%program
host=%host srcip=%srcip username=%username - successful login after brute force"; \
        shellcmd (logger -t SEC-SECURITY -p local0.crit "CREDENTIAL_COMPROMISE_SUSPECTED
program=\"%program\" host=\"%host\" srcip=\"%srcip\" username=\"%username\" - POSSIBLE ACCOUNT
COMPROMISE - IMMEDIATE ACTION REQUIRED"); \
        delete FAILED_LOGIN_ACTIVE_%username_%srcip
```

### LogZilla Trigger: Compromise Response

```
name: "Credential Compromise Response"
filter:
  - field: program
    op: eq
    value: SEC-SECURITY
  - field: message
    op: "=~"
    value: "CREDENTIAL_COMPROMISE_SUSPECTED"
actions:
  exec_script: true
  script_path: "/usr/local/bin/credential-compromise-response.sh"
  send_email: true
  send_email_template: |
    Subject: CRITICAL: Potential Credential Compromise

    User: {{:Username}}
    Source IP: {{:SrcIP}}
    Target Host: {{:host}}

    Successful login detected after brute force attack.
    Immediate investigation required.
```

## Cross-Service Brute Force Detection

### Multi-Service Attack Correlation

Detect attackers attempting brute force across multiple services (SSH, RDP, HTTP).

### SEC Rule: Cross-Service Correlation

```
# /etc/logzilla/sec/linux-auth/cross-service-correlation.sec
# ============================================
# Cross-Service Brute Force Detection
# Tracks failed login attempts across different services from same IP
# Detects when attacker tries multiple services (SSH, RDP, HTTP, etc.)
# ============================================
```

```
type=SingleWithThreshold
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="false"
desc=Cross-service attack detected from $3
thresh=15
window=900
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
       write /var/log/logzilla/sec/security.log "CROSS_SERVICE_ATTACK %t srcip=%srcip
program=%program host=%host username=%username total_attempts=15 window=900s - attacker targeting
multiple services"; \
       shellcmd logger -t SEC-SECURITY -p local0.alert "CROSS_SERVICE_ATTACK srcip=\"%srcip\"
program=\"%program\" host=\"%host\" username=\"%username\" total_attempts=15 window=900s - attacker
targeting multiple services"
```

# Time-Based Attack Pattern Detection

## Off-Hours Attack Detection

Detect brute force attacks occurring during unusual hours for additional context.

## SEC Rule: Time-Based Correlation

```
# /etc/logzilla/sec/linux-auth/time-based-correlation.sec
# ============================================
# Time-Based Brute Force Detection
# Enhanced brute force detection with time context
# Identifies attacks during off-hours or weekends as higher priority
# ============================================

type=SingleWithThreshold
ptype=RegExp
pattern=.*\[LINUX_AUTH\]\s+program="([^"]+)"\s+host="([^"]+)"\s+srcip="([^"]+)"\s+username="
([^"]+)"\s+auth_success="false"
desc=Brute force attack with time analysis for $4 from $3
thresh=8
window=300
action=eval %program ( "$1" ); \
       eval %host ( "$2" ); \
       eval %srcip ( "$3" ); \
       eval %username ( "$4" ); \
       eval %hour ( ( localtime( time() ) )[2] ); \
       eval %dow_raw ( ( localtime( time() ) )[6] ); \
       eval %day_of_week ( ( %dow_raw == 0 ) ? 7 : %dow_raw ); \
```

```
        eval %time_context ( ( ( %hour < 6 ) || ( %hour > 22 ) || ( %day_of_week > 5 ) ) ?
"suspicious" : "normal" ); \
        write /var/log/logzilla/sec/security.log "TIME_BASED_BRUTE_FORCE %t program=%program
host=%host srcip=%srcip username=%username attempts=8 time_context=%time_context hour=%hour
day=%day_of_week"; \
        shellcmd logger -t SEC-SECURITY -p local0.alert "BRUTE_FORCE_DETECTED program=\"%program\"
host=\"%host\" srcip=\"%srcip\" username=\"%username\" attempts=8 time_context=\"%time_context\"
hour=\"%hour\" day_of_week=\"%day_of_week\""
```

## Related Topics

- SOAR Security Orchestration (https://www.logzilla.ai/docs/event-correlation/soar-security-orchestration)
- Event Correlation Rule Types (https://www.logzilla.ai/docs/event-correlation/event-correlation-rule-types)
- Creating Triggers (https://www.logzilla.ai/docs/creating-triggers/)
- Advanced Event Correlation Walkthrough (https://www.logzilla.ai/docs/event-correlation/advanced-event-correlation-walkthrough)