LOGZILLA DOCUMENTATION

# Correlating Windows Events

Correlate Windows Security events 4624, 4625, 4732, and 1102 in LogZilla to detect successful brute-force attacks, privilege escalation, and log tampering

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

# Windows Event Correlation for Administrators

Windows administrators face unique challenges monitoring enterprise environments with thousands of events daily. LogZilla's Windows correlation capabilities combine pre-built triggers for immediate alerts with SEC correlation for complex attack pattern detection.

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).

# Brute Force Attack Detection

## Business Problem

Failed login alerts (Event ID 4625) generate thousands of events daily in enterprise environments. Simple triggers create alert fatigue, while sophisticated attacks go undetected.

## Correlation Solution

Detect successful brute force attacks by correlating failed attempts with subsequent successful logins.

## LogZilla Forwarder Configuration

**Required App**: `ms_windows` app (for `MSWin EventID`, `MSWin Failed Login User`, and `MSWin Failed Login Source Network` user tags)

**File**: `/etc/logzilla/forwarder.d/windows-brute-force.yaml`

```
# /etc/logzilla/forwarder.d/windows-brute-force.yaml
window_size: 0
type: sec
sec_name: windows-security
match:
  - field: MSWin EventID
    value: ["4624", "4625", "4732", "1102"]
rules:
- rewrite:
    message: >-
      [WINDOWS_AUTH] host=${:host} MSWin_EventID=${:MSWin EventID} SrcIP=${:SrcIP}
      Subject_User_Name=${:Subject User Name} Target_Domain_Name=${:Target Domain Name}
      Target_Group_Name=${:Target Group Name} Target_User_Name=${:Target User Name}
      Username=${:Username} User_Domain_Name=${:User Domain Name} $MESSAGE
```

## SEC Rule: Brute Force Detection

**File**: `/etc/logzilla/sec/windows-security/brute-force.sec`

New:

```
# =========================
# Windows brute-force (per user) - EventGroup variant
# =========================

# 1) Track failed logins per (host, srcip, user)
type=EventGroup
ptype=RegExp
# Allow newlines between fields
pattern=.*\[WINDOWS_AUTH\][\s\S]*host=([^\s"]+)[\s\S]*MSWin_EventID=4625[\s\S]*SrcIP=([^\s"]+)
[\s\S]*Username=([^\s"]+)
thresh=10
window=300
multact=no
# Group key is the desc (one group per unique values)
desc=Windows failed logon burst for user $3 from $2 to $1

# Fire when threshold is reached
action=eval %target_host ( "$1" ); \
       eval %client_ip ( "$2" ); \
       eval %user_lc ( lc("$3") ); \
       create BRUTE_FORCE_ACTIVE_%user_lc 1800; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit \
         "WINDOWS_BRUTE_FORCE user=\"%user_lc\" host=\"%target_host\" src=\"%client_ip\"")

# Optional: set vars when the group starts; clean up when it expires
init=eval %target_host ( "$1" ); eval %client_ip ( "$2" ); eval %user_lc ( lc("$3") )
end=delete BRUTE_FORCE_ACTIVE_%user_lc

# 2) Detect successful login after failures
type=Single
ptype=RegExp
pattern=.*\[WINDOWS_AUTH\][\s\S]*host=([^\s"]+)[\s\S]*MSWin_EventID=4624[\s\S]*SrcIP=([^\s"]+)
[\s\S]*Username=([^\s"]+)
desc=Windows successful logon after failures (same user) $3
action=eval %target_host ( "$1" ); \
       eval %client_ip ( "$2" ); \
       eval %user_lc ( lc("$3") ); \
       if (BRUTE_FORCE_ACTIVE_%user_lc) \
         shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit \
           "WINDOWS_COMPROMISE_SUSPECTED user=\"%user_lc\" host=\"%target_host\"
src=\"%client_ip\""); \
       delete BRUTE_FORCE_ACTIVE_%user_lc
```

Old:

```
# =========================
# Windows brute-force (per user, case-insensitive) - ALTERNATIVE SOLUTION
# =========================

# 1) Track failed logins per user using EventGroup
type=EventGroup
ptype=RegExp
pattern=.*\[WINDOWS_AUTH\]\s+host=([^\s"]+)\s+MSWin_EventID=4625\s+SrcIP=([^\s"]+)\s+Username=
([^\s"]+).*
thresh=10
desc=Windows failed logon burst for user $3 from $2 to $1
action=eval %target_host ( "$1" ); \
       eval %client_ip ( "$2" ); \
       eval %user_lc ( lc("$3") ); \
       create BRUTE_FORCE_ACTIVE_%user_lc 1800; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit \
         "WINDOWS_COMPROMISE_SUSPECTED target=\"%target_host\" src_ip=\"%src_ip\"
username=\"%user_lc\""); \

init=eval %target_host ( "$1" ); eval %client_ip ( "$2" ); eval %user_lc ( lc("$3") )
end=delete BRUTE_FORCE_ACTIVE_%user_lc
window=300
multact=no

# 2) Detect successful login after failures
type=Single
ptype=RegExp
pattern=.*\[WINDOWS_AUTH\]\s+host=([^\s"]+)\s+MSWin_EventID=4624\s+SrcIP=([^\s"]+)\s+Username=
([^\s"]+).*
desc=Windows successful logon after failures (same user) $3
action=eval %target_host ( "$1" ); \
       eval %client_ip ( "$2" ); \
       eval %user_lc ( lc("$3") ); \
       if (BRUTE_FORCE_ACTIVE_%user_lc) \
         shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit \
           "WINDOWS_COMPROMISE_SUSPECTED user=\"%user_lc\" host=\"%target_host\"
src=\"%client_ip\""); \
       delete BRUTE_FORCE_ACTIVE_%user_lc; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit \
         "WINDOWS_COMPROMISE_SUSPECTED target=\"%target_host\" src_ip=\"%src_ip\"
username=\"%user_lc\""); \
```

## LogZilla Trigger: Compromise Response

```
name: "Windows Account Compromise Response"
filter:
  - field: program
    op: eq
    value: SEC-WINDOWS-SECURITY
  - field: message
```

```
    op: "=~"
    value: "WINDOWS_COMPROMISE_SUSPECTED"
actions:
  exec_script: true
  script_path: "/usr/local/bin/windows-compromise-response.sh"
  send_email: true
  send_email_template: |
    Subject: CRITICAL: Windows Account Compromise

    User: {{event:ut:username}}
    Source IP: {{event:ut:SrcIP}}
    Target Host: {{event:ut:target}}

    Successful login detected after brute force attack.
    Account may be compromised - immediate investigation required.
```

## Intelligent Response Script

```
#!/bin/bash
# /usr/local/bin/windows-compromise-response.sh
# Called by SEC shellcmd - receives data via command-line arguments

USERNAME="$1"
SRC_IP="$2"
TARGET_HOST="$3"

# Query Active Directory for account details
ACCOUNT_TYPE=$(ldapsearch -x -h dc.company.com -b "dc=company,dc=com" \
               "(sAMAccountName=$USERNAME)" memberOf | grep -c "Domain Admins")

# Check IP reputation
IP_REPUTATION=$(curl -s "https://threat-intel.company.com/ip/$SRC_IP")

if [[ "$ACCOUNT_TYPE" -gt 0 ]]; then
    # Domain admin account compromised - immediate lockdown
    logger -t SECURITY-RESPONSE "Domain admin compromise: $USERNAME"

    # Disable account immediately
    net user "$USERNAME" /active:no /domain

    # Create critical incident
    curl -X POST "https://servicedesk.company.com/api/incidents" \
         -d "priority=critical&subject=Domain Admin Compromise&user=$USERNAME"

    # Alert CISO immediately
    curl -X POST "https://slack.company.com/api/webhooks/ciso-alerts" \
         -d "text=CRITICAL: Domain admin $USERNAME compromised from $SRC_IP"
else
    # Standard user account
    logger -t SECURITY-RESPONSE "User account compromise: $USERNAME"
```

```
    # Force password reset
    net user "$USERNAME" /passwordreq:yes /domain

    # Create high-priority ticket
    curl -X POST "https://servicedesk.company.com/api/tickets" \
         -d "priority=high&subject=Account Compromise&user=$USERNAME"
fi
```

# Privilege Escalation Detection

## Attack Pattern

Attackers often add compromised accounts to privileged groups, then clear audit logs to hide evidence.

## SEC Rule: Privilege Escalation Campaign

```
# Track additions to local groups (parse from message)
type=Single
ptype=RegExp
pattern=(?s)\[WINDOWS_AUTH\].*?host=([^\s]+).*?MSWin_EventID=4732.*?Target_User_Name=([^\s]+).*?
Username=([^\s]+)
desc=User added to local group user=$3 group=$2 host=$1
action=create PRIV_ESC_HOST_$1 3600; \
       write /var/log/logzilla/sec/windows-security-messages.log \
         "PRIVILEGE_GRANTED %t user=$3 group=$2 host=$1"; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.warning "PRIVILEGE_GRANTED user=\"$3\"
group=\"$2\" host=\"$1\"")

# Detect audit log clearing after privilege escalation (correlate on host)
type=Single
ptype=RegExp
pattern=(?s)\[WINDOWS_AUTH\].*?host=([^\s]+).*?MSWin_EventID=1102\b
context=PRIV_ESC_HOST_$1
desc=Audit log cleared after privilege escalation host=$1
action=write /var/log/logzilla/sec/windows-security-messages.log \
         "PRIVILEGE_ESCALATION_ATTACK %t host=$1 (1102 after 4732)"; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit "PRIVILEGE_ESCALATION_ATTACK
host=\"$1\" correlation=\"1102_after_4732\""); \
       delete PRIV_ESC_HOST_$1
```

# Service Flapping Detection

## Operational Problem

Critical Windows services that repeatedly crash indicate system instability but generate too many individual alerts to be useful.

## SEC Rule: Service Stability Monitoring

```
# Detect repeated service failures (7031) using regex-parsed service name
type=SingleWithThreshold
ptype=RegExp
pattern=(?s)\[WINDOWS_SERVICE\].*?MSWin_EventID=7031.*?Service_Name=\"([^\"]+)\".*?terminated
unexpectedly
desc=Service flapping detected service="$1"
thresh=5
window=600
action=write /var/log/logzilla/sec-windows-service-messages.log \
  SWT-HIT %t service="$1"; \
       write /var/log/logzilla/sec-windows-service-messages.log \
         SERVICE-FLAPPING 7031 %t service="$1" failures="5"
  shellcmd (logger -t SEC-WINDOWS-OPERATIONS -p local0.warning "SERVICE_FLAPPING service=\"$1\"
failures=\"5\"")
```

# Advanced Windows Correlation Patterns

## Malware Persistence Detection

Correlate suspicious process creation with service installation and network connections to detect malware establishing persistence.

```
# Track suspicious process execution from user directories (4688 comes first)
type=Single
ptype=RegExp
pattern=\[WINDOWS_PROCESS\].*?MSWin_EventID=4688.*?Process_Name="([^"]+)".*?
Process_Path="C:\\Users\\.*?\\([^"\\]+)".*?Username="([^"]*)".*?User_Domain_Name="([^"]*)"
desc=Suspicious process from user directory filename=$2 account=$4\\$3
action=eval %proc_user ( "$3" ); \
       eval %proc_domain ( "$4" ); \
       eval %proc_name ( "$1" ); \
       create SUSPICIOUS_PROC_$2 1800; \
       write /var/log/logzilla/sec/windows-process.log \
         SUSPICIOUS-PROCESS %t filename="$2" process_name="$1" account="$4\\$3"; \
       shellcmd (logger -t SEC-WINDOWS-MALWARE "SUSPICIOUS_PROCESS filename=\"$2\"
process_name=\"$1\" account=\"$4\\\\$3\"")
```

```
# Correlate with service installation (7045) to detect malware persistence
type=Single
ptype=RegExp
pattern=\[WINDOWS_PROCESS\].*?MSWin_EventID=7045.*?Service_Account="([^"]*)".*?
Service_Filename="C:\\Users\\.*?\\([^"\\]+)".*?Service_Name="([^"]+)"
context=SUSPICIOUS_PROC_$2
desc=Malware persistence detected - process to service service=$3 filename=$2 svc_account=$1
action=write /var/log/logzilla/sec/windows-process.log \
         MALWARE-PERSISTENCE %t service="$3" filename="$2" service_account="$1"
process_account="%proc_domain\\%proc_user" process_name="%proc_name"; \
       shellcmd (logger -t SEC-WINDOWS-SECURITY -p local0.crit "MALWARE_PERSISTENCE service=\"$3\"
filename=\"$2\" service_account=\"$1\" process_account=\"%proc_domain\\\\%proc_user\"")
```

## PowerShell Attack Chain Detection

Detect PowerShell-based attacks by correlating execution with credential access and network activity.

```
# Track PowerShell execution
type=Single
ptype=RegExp
pattern=\[WINDOWS_PROCESS\][\s\S]*?MSWin_EventID=4104[\s\S]*?Script_Path="([^"]*)"[\s\S]*?
User_Domain_Name="[^"]*"\s*([\s\S]*)
desc=PowerShell script execution detected
action=eval %ps_script_path ( "$1" ); \
       eval %ps_message ( "$2" ); \
       create POWERSHELL_ACTIVE_%ps_script_path 900; \
       eval %cred_access ( ( %ps_message =~ /(Invoke-Mimikatz|Get-Credential|ConvertTo-
SecureString)/i ) ? 1 : 0 ); \
       shellcmd ([ %cred_access -eq 1 ] && logger -t SEC-WINDOWS-SECURITY -p local0.alert
"POWERSHELL_CREDENTIAL_ACCESS script_path=\"%ps_script_path\"" || true)
```

## Related Topics

- SOAR Security Orchestration (https://www.logzilla.ai/docs/event-correlation/soar-security-orchestration)
- Brute Force Attack Detection (https://www.logzilla.ai/docs/event-correlation/brute-force-attack-detection)
- Event Correlation Rule Types (https://www.logzilla.ai/docs/event-correlation/event-correlation-rule-types)
- Windows Event Forwarding (https://www.logzilla.ai/docs/receiving-data/windows-event-forwarding)
- Creating Triggers (https://www.logzilla.ai/docs/creating-triggers/)