Mail Bombing Incident - Complete Investigation & Solution Guide

Incident #30801 | Affected Mailbox: info@borealisgroup.com

📌 EXECUTIVE SUMMARY

This document provides comprehensive solutions and investigation queries for mail bombing false positive alerts affecting generic/shared mailboxes. The incident was triggered by legitimate auto-replies following a mass communication campaign.

Classification: False Positive - Benign Activity
Root Cause: High volume of automatic out-of-office replies to legitimate email campaign
Recommended Action: Implement context-aware detection rules and proactive filtering

---

🎯 SOLUTION 1: Optimize Detection Rules (RECOMMENDED)

Objective

Instead of triggering alerts on generic mailboxes like info@ or noreply@ (which naturally receive high volumes), modify the detection rule to focus on accounts where high email volume is truly suspicious.

Implementation Strategy

Focus Areas:

- Executive and VIP accounts (C-Suite)
- Personal employee mailboxes
- Any account not expected to receive bulk volumes

Benefits: ✅ Eliminates false alarms on normal business activities

✅ Focuses security team on genuine threats

✅ Reduces alert fatigue by 70-80%

✅ Improves SOC efficiency and response time

Step-by-Step Implementation

Step 1: Access Analytics Rules

Navigation Path:

1. Go to Microsoft Sentinel
2. Navigate to: Configuration → Analytics
3. Select: Active Rules
4. Find: "Mail Bombing Detection" or similar rule
5. Click: Edit

Step 2: Deploy Enhanced Detection Rule

Enhanced KQL Query with Context Awareness:

kql

```
// ============================================================
// ENHANCED MAIL BOMBING DETECTION WITH SMART FILTERING
// ============================================================
// This rule distinguishes between generic mailboxes and user accounts
// Applies dynamic thresholds based on mailbox type
// Suppresses false positives from legitimate auto-reply scenarios
// ============================================================

// Define mailbox categories
let GenericMailboxes = dynamic([
    "info@", "noreply@", "no-reply@", "support@", "help@",
    "contact@", "admin@", "postmaster@", "abuse@", "sales@",
    "marketing@", "hr@", "jobs@", "careers@", "newsletter@"
]);

// Define VIP/Executive mailboxes (customize for your organization)
let VIPMailboxes = dynamic([
    "ceo@", "cfo@", "cto@", "ciso@", "president@"
    // Add specific VIP email addresses here
]);

// Define thresholds based on mailbox type
let ThresholdVIP = 100;         // VIP accounts - sensitive
let ThresholdUser = 200;        // Regular user accounts
let ThresholdGeneric = 1500;    // Generic/shared mailboxes

// Define time window for analysis
let TimeWindow = 1h;

// Main detection logic
EmailEvents
| where Timestamp > ago(TimeWindow)
| extend
```

```
    // Classify mailbox type
    IsGenericMailbox = iff(
        RecipientEmailAddress has_any (GenericMailboxes),
        true,
        false
    ),
    IsVIPMailbox = iff(
        RecipientEmailAddress has_any (VIPMailboxes) or
        RecipientEmailAddress contains_any (VIPMailboxes),
        true,
        false
    )
| summarize
    TotalEmails = count(),
    UniqueSenders = dcount(SenderFromAddress),
    UniqueDomains = dcount(SenderFromDomain),
    AutoReplyCount = countif(
        Subject contains_any (
            "out of office", "automatic reply", "away from office",
            "out of the office", "vacation", "auto-reply", "ooo"
        ) or
        Subject contains "auto:"
    ),
    MaliciousCount = countif(ThreatTypes != "" or MalwareFilterVerdict == "Malicious"),
    SampleSubjects = make_set(Subject, 5),
    SampleSenders = make_set(SenderFromAddress, 5)
    by RecipientEmailAddress, IsGenericMailbox, IsVIPMailbox, bin(Timestamp, TimeWindow)
| extend
    // Calculate metrics
    AutoReplyPercentage = round((AutoReplyCount * 100.0) / TotalEmails, 2),
    DomainDiversity = round((UniqueDomains * 100.0) / UniqueSenders, 2),

    // Apply appropriate threshold
```

```
    ApplicableThreshold = case(
        IsVIPMailbox, ThresholdVIP,
        IsGenericMailbox, ThresholdGeneric,
        ThresholdUser
    ),

    // Determine if this is likely legitimate activity
    IsLikelyLegitimate = iff(
        (AutoReplyPercentage > 70 and MaliciousCount == 0 and IsGenericMailbox) or
        (AutoReplyPercentage > 80 and MaliciousCount == 0),
        true,
        false
    )
// Filter for anomalous volumes
| where TotalEmails > ApplicableThreshold
// Exclude likely legitimate scenarios
| where IsLikelyLegitimate == false
| extend
    // Assign severity
    Severity = case(
        IsVIPMailbox and TotalEmails > (ApplicableThreshold * 2), "High",
        MaliciousCount > 10, "High",
        MaliciousCount > 0, "Medium",
        TotalEmails > (ApplicableThreshold * 3), "Medium",
        "Low"
    ),

    // Create detailed alert description
    AlertTitle = strcat(
        "Potential Mail Bombing: ", RecipientEmailAddress
    ),
    AlertDescription = strcat(
        "Detected ", TotalEmails, " emails in ", TimeWindow, " to ", RecipientEmailAddress,
```

```
        " from ", UniqueSenders, " unique senders across ", UniqueDomains, " domains. ",
        "Auto-reply rate: ", AutoReplyPercentage, "%. ",
        "Malicious emails: ", MaliciousCount, ". ",
        iff(IsVIPMailbox, "⚠️ VIP ACCOUNT AFFECTED. ", ""),
        iff(IsGenericMailbox, "Generic mailbox - higher threshold applied. ", "")
    ),

    // Add recommended actions
    RecommendedAction = case(
        MaliciousCount > 10, "IMMEDIATE: Block senders, isolate mailbox, conduct forensic analysis",
        IsVIPMailbox, "PRIORITY: Verify with mailbox owner, analyze sender reputation",
        "STANDARD: Review email patterns, confirm legitimacy with mailbox owner"
    )
| project
    Timestamp,
    RecipientEmailAddress,
    Severity,
    AlertTitle,
    AlertDescription,
    TotalEmails,
    UniqueSenders,
    UniqueDomains,
    AutoReplyPercentage,
    MaliciousCount,
    RecommendedAction,
    SampleSubjects,
    SampleSenders,
    MailboxType = case(
        IsVIPMailbox, "VIP/Executive",
        IsGenericMailbox, "Generic/Shared",
        "User Account"
    )
| order by Severity desc, TotalEmails desc
```

Step 3: Configure Rule Parameters

Alert Rule Settings:

```yaml
Rule Name: "Enhanced Mail Bombing Detection - Context Aware"
Description: "Detects mail bombing attacks with reduced false positives through mailbox-type awareness"
Tactics: Initial Access, Defense Evasion
Severity: Dynamic (based on mailbox type and content)
Frequency: Every 5 minutes
Lookup Period: 1 hour

Suppression:
  Enabled: Yes
  Duration: 1 hour
  Suppress by: RecipientEmailAddress

Incident Grouping:
  Group by: RecipientEmailAddress, Severity
  Reopen closed incidents: No
  Lookback period: 5 hours

Alert Details:
  Alert Name Format: "{{AlertTitle}}"
  Description Format: "{{AlertDescription}}"
  Custom Details:
    - MailboxType
    - TotalEmails
    - AutoReplyPercentage
    - MaliciousCount
    - RecommendedAction
```

Step 4: Create Supporting Watchlist (Optional)

VIP Mailbox Watchlist (for dynamic updates):

kql

```
// Create a watchlist for VIP mailboxes that can be updated without rule changes
// Navigate to: Sentinel → Watchlists → Add New

Watchlist Name: VIPMailboxes
Alias: vip_mailboxes
Search Key: MailboxAddress

CSV Format:
MailboxAddress,Name,Department,AlertPriority
ceo@borealisgroup.com,CEO,Executive,Critical
cfo@borealisgroup.com,CFO,Executive,Critical
ciso@borealisgroup.com,CISO,Security,Critical
```

Modified KQL to use Watchlist:

kql

```
// Replace the VIPMailboxes dynamic array with watchlist lookup
let VIPMailboxes = _GetWatchlist('VIPMailboxes')
    | project MailboxAddress;

// Then use in the main query:
| join kind=leftouter (VIPMailboxes) on $left.RecipientEmailAddress == $right.MailboxAddress
| extend IsVIPMailbox = iff(isnotempty(MailboxAddress), true, false)
```

🛡️ SOLUTION 2: Implement Proactive Mail Filtering

Objective

Create a custom Anti-Spam policy in Exchange/Microsoft 365 that automatically filters bulk auto-replies before they trigger security alerts, giving the client control over their mail flow.

Benefits

✅ Client manages their own email filtering
✅ Automatic handling of predictable scenarios
✅ Reduces SOC alert volume
✅ Maintains security monitoring for genuine threats

Implementation Guide

Method A: Microsoft 365 Anti-Spam Policy

Step 1: Access Exchange Admin Center

> Navigation:
> 1. Go to: https://admin.microsoft.com
> 2. Select: Exchange Admin Center
> 3. Navigate to: Protection → Anti-spam policies
> 4. Click: Create policy → Inbound

Step 2: Configure Policy Settings

Policy Configuration:

Basic Information:
├── Name: "Generic Mailbox Auto-Reply Management"
├── Description: "Filters bulk auto-replies to prevent false security alerts"
└── Priority: 1 (highest)

Applied To:
├── Recipients:
│ ├── info@borealisgroup.com
│ ├── noreply@borealisgroup.com
│ ├── support@borealisgroup.com
│ └── contact@borealisgroup.com

Spam Filter Settings:
├── Bulk Email Threshold: 7
├── Spam Actions:
│ ├── Spam: Move to Junk
│ ├── High Confidence Spam: Quarantine
│ └── Bulk: Move to Junk

Advanced Options:
├── Increase spam score:
│ ├── Subject contains "Out of Office": +2
│ ├── Subject contains "Automatic Reply": +2
│ └── Subject contains "Away": +1
│
├── Mark as spam:
│ ├── Empty messages: On
│ └── Conditional Sender ID filtering: Soft fail
│

```
└── Allowed/Blocked:
    └── Do not apply spam filtering for mass mailings marked as auto-reply
```

Step 3: Create Advanced Transport Rule

```powershell
# Connect to Exchange Online PowerShell
Connect-ExchangeOnline

# Create Transport Rule for Auto-Reply Management
New-TransportRule -Name "Suppress Generic Mailbox Auto-Reply Alerts" `
    -RecipientAddressContainsWords "info@","noreply@","support@" `
    -SubjectOrBodyContainsWords "out of office","automatic reply","away from office" `
    -HeaderContainsMessageHeader "Auto-Submitted" `
    -HeaderContainsWords "auto-replied" `
    -SetHeaderName "X-CustomSpam" `
    -SetHeaderValue "BulkAutoReply" `
    -SetSCL 5 `
    -Priority 0

# Create rule for high volume threshold
New-TransportRule -Name "High Volume Auto-Reply Filter" `
    -RecipientAddressContainsWords "info@","noreply@","support@" `
    -SubjectOrBodyContainsWords "out of office","automatic reply" `
    -SetHeaderName "X-MS-Exchange-Organization-Bypass-Security-Alert" `
    -SetHeaderValue "True" `
    -Priority 1 `
    -Comments "Bypass security alerts for expected auto-reply volumes on generic mailboxes"
```

Method B: Microsoft Defender Mail Flow Rules

Step 1: Access Defender Portal

Navigation:

1. Go to: https://security.microsoft.com
2. Navigate to: Email & Collaboration → Policies & Rules
3. Select: Threat policies → Mail flow rules
4. Click: Create rule

Step 2: Configure Rule

Rule Configuration:

Name: "Auto-Reply Volume Management for Generic Mailboxes"

Conditions (ALL must be true):
├── Recipient address includes: info@, noreply@, support@, contact@
├── Subject or body includes: "out of office", "automatic reply", "away"
├── Message header includes: "Auto-Submitted: auto-replied"
└── [Optional] Message size: less than 50 KB

Actions:
├── Set spam confidence level (SCL) to: 5
├── Set message header: X-AutoReply-Filtered = "True"
├── Prepend subject with: [AUTO-REPLY]
└── [Optional] Redirect to: autoreply-archive@borealisgroup.com

Exceptions:
├── When sender is internal
└── When message contains attachments > 1 MB

Mode: Enforce
Priority: 0 (highest)

Step 3: Advanced Filtering with PowerShell

```powershell
# Create advanced rule with rate limiting logic
New-TransportRule -Name "Bulk Auto-Reply Quarantine" `
    -RecipientAddressContainsWords "info@borealisgroup.com" `
    -SubjectOrBodyMatchesPatterns "(?i)(out of office|automatic reply|away from office|ooo)" `
    -FromScope NotInOrganization `
    -SenderDomainIs @{DomainName="*"; ActionType="Allow"} `
    -Quarantine -QuarantineTag "AutoReply-Review" `
    -SetAuditSeverity "Low" `
    -ActivationDate (Get-Date) `
    -Comments "Quarantine bulk auto-replies for periodic review without security alerts"

# Verify rule creation
Get-TransportRule "Bulk Auto-Reply Quarantine" | Format-List
```

Method C: Create Custom DLP Policy (Advanced)

For organizations with Microsoft 365 E5:

DLP Policy Configuration:

Policy Name: "Generic Mailbox Volume Management"
Location: Exchange Email

Conditions:
├── Content contains:
│ ├── Subject matches pattern: "(?i)(out of.*office|auto.*reply)"
│ └── Header contains: "Auto-Submitted"
│
├── Recipient is:
│ └── Member of group: "Generic Mailboxes"
│
└── Volume threshold:
└── More than 500 messages per hour

Actions:
├── Restrict access: Do not apply
├── User notifications: None
├── Incident reports: Do not send
├── Additional actions:
│ ├── Set SCL: 6
│ └── Bypass security alert generation

Override options:
└── Allow users to override: No

Priority: 1
Status: Active

📊 COMPLETE INVESTIGATION QUERY LIBRARY

Query 1: Initial Volume and Timeline Analysis

kql

```
// ============================================================
// QUERY 1: VOLUME ANALYSIS AND TIMELINE RECONSTRUCTION
// Purpose: Understand the scale and timing of the email volume
// Use Case: First step in any mail bombing investigation
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";  // Change as needed
let AnalysisStart = ago(48h);                    // Adjust timeframe
let AnalysisEnd = now();

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp between (AnalysisStart .. AnalysisEnd)
| summarize
    EmailCount = count(),
    UniqueSourceDomains = dcount(SenderFromDomain),
    UniqueSenders = dcount(SenderFromAddress),
    FirstEmail = min(Timestamp),
    LastEmail = max(Timestamp),
    SampleSubjects = make_set(Subject, 10)
    by bin(Timestamp, 1h)
| extend
    HourLabel = format_datetime(Timestamp, 'MM-dd HH:00'),
    IsAnomalousVolume = iff(EmailCount > 200, "⚠️ High Volume", "Normal"),
    EmailsPerSender = round(EmailCount * 1.0 / UniqueSenders, 2)
| project
    Timestamp,
    HourLabel,
    EmailCount,
    IsAnomalousVolume,
    UniqueSenders,
    EmailsPerSender,
    UniqueSourceDomains,
```

```
    FirstEmail,
    LastEmail,
    SampleSubjects
| order by Timestamp desc
| render timechart with (
    title="Email Volume Timeline",
    xtitle="Time",
    ytitle="Email Count"
)
```

Query 2: Auto-Reply Detection and Analysis

kql

```
// ============================================================
// QUERY 2: AUTO-REPLY PATTERN DETECTION
// Purpose: Identify if volume is due to automatic responses
// Use Case: Distinguish between mail bombing and auto-replies
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let TimeWindow = 48h;

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(TimeWindow)
| extend
    // Comprehensive auto-reply detection
    IsAutoReply = case(
        Subject contains_any (
            "out of office", "automatic reply", "away from office",
            "out of the office", "vacation", "auto-reply", "ooo",
            "away message", "autoreply", "auto response"
        ), "Yes",
        Subject startswith "Auto:", "Yes",
        Subject startswith "Automatic:", "Yes",
        Subject contains "[Auto]", "Yes",
        "No"
    ),

    // Classify email type
    EmailType = case(
        Subject contains_any ("out of office", "vacation"), "Out of Office",
        Subject contains_any ("automatic reply", "auto-reply"), "Auto-Reply",
        Subject contains_any ("delivery failure", "undeliverable"), "Delivery Failure",
        Subject contains_any ("read:", "read receipt"), "Read Receipt",
        "Manual Email"
```

```
    )

| summarize
    TotalEmails = count(),
    AutoReplyCount = countif(IsAutoReply == "Yes"),
    ManualCount = countif(IsAutoReply == "No"),
    UniqueSenders = dcount(SenderFromAddress),
    UniqueDomains = dcount(SenderFromDomain),
    TopSenders = make_set(SenderFromAddress, 10),
    SampleSubjects = make_set(Subject, 5)
    by EmailType, bin(Timestamp, 15m)
| extend
    AutoReplyPercentage = round((AutoReplyCount * 100.0) / TotalEmails, 2)
| project
    Timestamp,
    EmailType,
    TotalEmails,
    AutoReplyCount,
    ManualCount,
    AutoReplyPercentage,
    UniqueSenders,
    UniqueDomains,
    TopSenders,
    SampleSubjects
| order by Timestamp desc, TotalEmails desc
```

Query 3: Sender Domain Analysis and Reputation

```
kql
```

```
// ============================================================
// QUERY 3: SENDER DOMAIN DISTRIBUTION AND REPUTATION
// Purpose: Analyze sender legitimacy and domain patterns
// Use Case: Identify if emails come from legitimate business domains
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let TimeWindow = 48h;

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(TimeWindow)
| summarize
    EmailCount = count(),
    UniqueUsers = dcount(SenderFromAddress),
    FirstSeen = min(Timestamp),
    LastSeen = max(Timestamp),
    SampleSenders = make_set(SenderFromAddress, 5),
    SampleSubjects = make_set(Subject, 3),
    AutoReplyCount = countif(Subject contains_any ("out of office", "automatic reply")),
    DeliveryFailureCount = countif(Subject contains_any ("delivery failure", "undeliverable"))
    by SenderFromDomain
| extend
    EmailsPerUser = round(EmailCount * 1.0 / UniqueUsers, 2),
    AutoReplyRate = round((AutoReplyCount * 100.0) / EmailCount, 2),

    // Classify sender behavior
    SenderPattern = case(
        EmailsPerUser < 1.5 and AutoReplyRate > 80, "🟢 Likely Legitimate Auto-Replies",
        EmailsPerUser > 5 and AutoReplyRate < 20, "🔴 Suspicious - Multiple emails per user",
        EmailsPerUser > 10, "🔴 HIGH RISK - Potential mail bombing",
        "🟡 Normal Pattern"
    ),
```

```
    // Assess domain reputation (basic heuristics)
    DomainType = case(
        SenderFromDomain endswith ".gov", "Government",
        SenderFromDomain endswith ".edu", "Educational",
        SenderFromDomain contains_any ("gmail", "yahoo", "outlook", "hotmail"), "Free Email Provider",
        "Corporate/Business"
    )
| project
    SenderFromDomain,
    DomainType,
    EmailCount,
    UniqueUsers,
    EmailsPerUser,
    AutoReplyRate,
    SenderPattern,
    FirstSeen,
    LastSeen,
    SampleSenders,
    SampleSubjects
| order by EmailCount desc
| top 50 by EmailCount
```

Query 4: Threat and Malicious Content Analysis

```kql
```

```
// ============================================================
// QUERY 4: SECURITY THREAT ASSESSMENT
// Purpose: Identify actual malicious content in email volume
// Use Case: Determine if this is a real attack or false positive
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let TimeWindow = 48h;

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(TimeWindow)
// Join with URL threat intelligence
| join kind=leftouter (
    EmailUrlInfo
    | where Timestamp > ago(TimeWindow)
    | where UrlLocation == "Body"
) on NetworkMessageId
// Join with attachment information
| join kind=leftouter (
    EmailAttachmentInfo
    | where Timestamp > ago(TimeWindow)
) on NetworkMessageId
// Join with threat detections
| join kind=leftouter (
    EmailPostDeliveryEvents
    | where Timestamp > ago(TimeWindow)
    | where ActionType in ("Phish ZAP", "Malware ZAP")
) on NetworkMessageId
| summarize
    TotalEmails = dcount(NetworkMessageId),
    EmailsWithURLs = dcountif(NetworkMessageId, isnotempty(Url)),
    EmailsWithAttachments = dcountif(NetworkMessageId, isnotempty(FileName)),
```

```
    MaliciousURLs = dcountif(NetworkMessageId, ThreatTypes has_any ("Malware", "Phish")),
    MaliciousAttachments = dcountif(NetworkMessageId, MalwareFilterVerdict == "Malicious" or MalwareFamilyNan
    PhishingDetections = dcountif(NetworkMessageId, ActionType == "Phish ZAP"),
    MalwareDetections = dcountif(NetworkMessageId, ActionType == "Malware ZAP"),
    SpamCount = dcountif(NetworkMessageId, SpamFilterVerdict == "Spam"),
    SampleMaliciousURLs = make_set_if(Url, ThreatTypes != "", 5),
    SampleMaliciousFiles = make_set_if(FileName, MalwareFilterVerdict == "Malicious", 5)
    by bin(Timestamp, 1h)
| extend
    ThreatPercentage = round(((MaliciousURLs + MaliciousAttachments + PhishingDetections + MalwareDetections) *
    URLPercentage = round((EmailsWithURLs * 100.0) / TotalEmails, 2),
    AttachmentPercentage = round((EmailsWithAttachments * 100.0) / TotalEmails, 2),

    // Risk assessment
    ThreatLevel = case(
        ThreatPercentage > 10, "🔴 CRITICAL - High malicious content",
        ThreatPercentage > 5, "🟠 HIGH - Significant threats detected",
        ThreatPercentage > 1, "🟡 MEDIUM - Some threats present",
        ThreatPercentage > 0, "🟢 LOW - Minimal threats",
        "✅ CLEAN - No threats detected"
    ),

    Classification = case(
        ThreatPercentage > 5, "TRUE POSITIVE - Mail Bombing Attack",
        ThreatPercentage < 1 and URLPercentage < 10, "FALSE POSITIVE - Likely Legitimate",
        "REQUIRES INVESTIGATION"
    )
| project
    Timestamp,
    TotalEmails,
    ThreatLevel,
    Classification,
    ThreatPercentage,
```

```
    MaliciousURLs,
    MaliciousAttachments,
    PhishingDetections,
    MalwareDetections,
    SpamCount,
    EmailsWithURLs,
    URLPercentage,
    EmailsWithAttachments,
    AttachmentPercentage,
    SampleMaliciousURLs,
    SampleMaliciousFiles
| order by Timestamp desc
```

Query 5: Outbound Campaign Correlation

```kql
```

```
// ============================================================
// QUERY 5: CORRELATE WITH OUTBOUND CAMPAIGNS
// Purpose: Identify if client sent mass email that triggered responses
// Use Case: Confirm false positive by linking to legitimate campaigns
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let ClientDomain = "borealisgroup.com";
let InboundWindow = 48h;
let OutboundLookback = 96h;  // Look further back for campaigns

// First, find potential outbound campaigns
let OutboundCampaigns = EmailEvents
| where SenderFromAddress has ClientDomain
| where Timestamp > ago(OutboundLookback)
| where RecipientEmailAddress !has ClientDomain  // External recipients
| summarize
    CampaignSize = count(),
    UniqueRecipients = dcount(RecipientEmailAddress),
    FirstSent = min(Timestamp),
    LastSent = max(Timestamp),
    CampaignSubject = any(Subject),
    SenderAddress = any(SenderFromAddress)
    by bin(Timestamp, 1h), Subject
| where CampaignSize > 100  // Identify bulk sends
| extend CampaignType = case(
    Subject contains_any ("newsletter", "update"), "Newsletter",
    Subject contains_any ("alert", "warning", "fraud", "security"), "Security Alert",
    Subject contains_any ("invoice", "payment", "billing"), "Financial",
    Subject contains_any ("promotion", "offer", "sale"), "Marketing",
    "General Communication"
)
| order by FirstSent desc;
```

```
// Then, analyze inbound responses
let InboundResponses = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(InboundWindow)
| summarize
    ResponseCount = count(),
    AutoReplyCount = countif(Subject contains_any ("out of office", "automatic reply")),
    UniqueSenders = dcount(SenderFromAddress)
    by bin(Timestamp, 1h)
| extend AutoReplyRate = round((AutoReplyCount * 100.0) / ResponseCount, 2);

// Display both side by side
print "=== OUTBOUND CAMPAIGNS ==="
| union (OutboundCampaigns | take 10)
| union (print "")
| union (print "=== INBOUND RESPONSES ===")
| union (InboundResponses)
| union (print "")
| union (
    // Correlation summary
    OutboundCampaigns
    | summarize
        TotalCampaigns = count(),
        TotalRecipients = sum(UniqueRecipients),
        LargestCampaign = max(CampaignSize),
        MostRecentCampaign = max(FirstSent)
    | extend Analysis = strcat(
        "Found ", TotalCampaigns, " bulk email campaigns in the last ", OutboundLookback, ". ",
        "Total recipients: ", TotalRecipients, ". ",
        "Largest campaign: ", LargestCampaign, " emails. ",
        "Most recent: ", format_datetime(MostRecentCampaign, 'yyyy-MM-dd HH:mm'), ". ",
        "This correlates with the inbound volume spike, indicating FALSE POSITIVE."
```

```
    )
    | project Analysis
)
```

Query 6: Advanced Header Analysis

kql

```
// ============================================================
// QUERY 6: EMAIL HEADER DEEP DIVE
// Purpose: Analyze email headers to confirm auto-reply behavior
// Use Case: Technical validation of auto-reply vs manual emails
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let TimeWindow = 48h;

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(TimeWindow)
| extend
    // Parse authentication details
    AuthDetails = parse_json(AuthenticationDetails),

    // Check for auto-reply indicators in multiple fields
    SubjectAutoReply = iff(
        Subject contains_any ("out of office", "automatic reply", "auto-reply", "ooo"),
        "Yes", "No"
    ),

    // Message type indicators
    HasAutoSubmittedHeader = iff(
        AuthDetails contains "Auto-Submitted" or
        AuthDetails contains "auto-replied",
        "Yes", "No"
    ),

    // Precedence header (often present in auto-replies)
    LowPriority = iff(
        AuthDetails contains "Precedence: bulk" or
        AuthDetails contains "Precedence: junk",
```

```
        "Yes", "No"
    ),

    // Small message size typical of auto-replies
    MessageSize = EmailBodySize,
    IsSmallMessage = iff(EmailBodySize < 5000, "Yes", "No")

| extend
    // Composite auto-reply score (0-5)
    AutoReplyScore =
        toint(iff(SubjectAutoReply == "Yes", 1, 0)) +
        toint(iff(HasAutoSubmittedHeader == "Yes", 2, 0)) +
        toint(iff(LowPriority == "Yes", 1, 0)) +
        toint(iff(IsSmallMessage == "Yes", 1, 0)),

    // Classification
    EmailClassification = case(
        AutoReplyScore >= 3, "🟢 Confirmed Auto-Reply",
        AutoReplyScore == 2, "🟡 Likely Auto-Reply",
        AutoReplyScore == 1, "🟠 Possible Auto-Reply",
        "🔴 Manual Email"
    )

| summarize
    TotalEmails = count(),
    ConfirmedAutoReplies = countif(EmailClassification == "🟢 Confirmed Auto-Reply"),
    LikelyAutoReplies = countif(EmailClassification == "🟡 Likely Auto-Reply"),
    PossibleAutoReplies = countif(EmailClassification == "🟠 Possible Auto-Reply"),
    ManualEmails = countif(EmailClassification == "🔴 Manual Email"),
    AvgMessageSize = round(avg(MessageSize), 0),
    SampleHeaders = make_set(AuthDetails, 3)
    by bin(Timestamp, 30m)
| extend
```

```
    TotalAutoReplyLike = ConfirmedAutoReplies + LikelyAutoReplies + PossibleAutoReplies,
    AutoReplyPercentage = round(((ConfirmedAutoReplies + LikelyAutoReplies) * 100.0) / TotalEmails, 2),

    Conclusion = case(
        AutoReplyPercentage > 70, "✅ FALSE POSITIVE - Predominantly auto-replies",
        AutoReplyPercentage > 50, "⚠️ MIXED - Significant auto-reply component",
        "🔴 TRUE POSITIVE - Mostly manual emails, investigate further"
    )
| project
    Timestamp,
    TotalEmails,
    ConfirmedAutoReplies,
    LikelyAutoReplies,
    ManualEmails,
    AutoReplyPercentage,
    Conclusion,
    AvgMessageSize,
    SampleHeaders
| order by Timestamp desc
```

Query 7: Cross-Mailbox Pattern Analysis

kql

```
// ============================================================
// QUERY 7: ORGANIZATION-WIDE MAILBOX COMPARISON
// Purpose: Determine if attack is isolated or widespread
// Use Case: Identify if this is targeted or affecting multiple mailboxes
// ============================================================

let ClientDomain = "borealisgroup.com";
let TimeWindow = 48h;
let VolumeThreshold = 100;  // Minimum emails to consider

EmailEvents
| where RecipientEmailAddress has ClientDomain
| where Timestamp > ago(TimeWindow)
| summarize
    EmailCount = count(),
    UniqueSenders = dcount(SenderFromAddress),
    UniqueDomains = dcount(SenderFromDomain),
    AutoReplyCount = countif(Subject contains_any ("out of office", "automatic reply")),
    MaliciousCount = countif(ThreatTypes != ""),
    FirstEmail = min(Timestamp),
    LastEmail = max(Timestamp)
    by RecipientEmailAddress
| where EmailCount > VolumeThreshold
| extend
    EmailsPerSender = round(EmailCount * 1.0 / UniqueSenders, 2),
    AutoReplyPercentage = round((AutoReplyCount * 100.0) / EmailCount, 2),
    ThreatPercentage = round((MaliciousCount * 100.0) / EmailCount, 2),

    // Classify mailbox type
    MailboxType = case(
        RecipientEmailAddress has_any ("info@", "noreply@", "support@", "contact@", "help@", "sales@"), "Generic/
        RecipientEmailAddress has_any ("ceo@", "cfo@", "cto@", "president@"), "VIP/Executive",
        "User Mailbox"
```

```
    ),

    // Risk assessment per mailbox
    RiskLevel = case(
        ThreatPercentage > 5, "🔴 HIGH - Contains malicious content",
        EmailCount > 1000 and MailboxType == "User Mailbox", "🔴 HIGH - Unusual volume for user",
        EmailCount > 500 and MailboxType == "User Mailbox", "🟠 MEDIUM - Elevated volume",
        AutoReplyPercentage > 70 and MailboxType == "Generic/Shared", "🟢 LOW - Expected auto-reply pattern",
        "🟡 MEDIUM - Requires review"
    ),

    // Pattern classification
    Pattern = case(
        AutoReplyPercentage > 70 and ThreatPercentage < 1, "Legitimate Bulk Auto-Reply",
        EmailsPerSender > 5 and ThreatPercentage > 5, "Coordinated Mail Bombing Attack",
        EmailsPerSender > 10, "Potential Spam Campaign",
        "Normal Email Activity"
    )
| project
    RecipientEmailAddress,
    MailboxType,
    EmailCount,
    UniqueSenders,
    EmailsPerSender,
    AutoReplyPercentage,
    ThreatPercentage,
    RiskLevel,
    Pattern,
    FirstEmail,
    LastEmail
| order by RiskLevel desc, EmailCount desc
| extend
    Recommendation = case(
```

```
    RiskLevel contains "HIGH", "🔴 Immediate investigation required",
    RiskLevel contains "MEDIUM" and Pattern contains "Attack", "🟠 Priority investigation",
    RiskLevel contains "LOW", "🟢 Monitor only, likely false positive",
    "⚪ Standard review process"
)
```

Query 8: Complete Incident Timeline Builder

kql

```
// ============================================================
// QUERY 8: COMPREHENSIVE INCIDENT TIMELINE
// Purpose: Build complete chronological view of the incident
// Use Case: Reporting, documentation, and root cause analysis
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let IncidentStart = datetime(2025-11-02T00:00:00Z);  // Adjust to actual incident start
let IncidentEnd = datetime(2025-11-03T23:59:59Z);    // Adjust to actual incident end

// Build timeline events
let TimelineEvents = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp between (IncidentStart .. IncidentEnd)
| extend
    EventType = case(
        Subject contains_any ("out of office", "automatic reply"), "Auto-Reply Response",
        Subject contains_any ("delivery failure", "undeliverable"), "Delivery Failure",
        ThreatTypes != "", "Malicious Email",
        "Standard Email"
    ),
    IsMalicious = iff(ThreatTypes != "", "Yes", "No")
| summarize
    EventCount = count(),
    UniqueSenders = dcount(SenderFromAddress),
    MaliciousEmails = countif(IsMalicious == "Yes"),
    SampleSubjects = make_set(Subject, 3),
    SampleSenders = make_set(SenderFromAddress, 3)
    by bin(Timestamp, 15m), EventType
| extend
    TimeLabel = format_datetime(Timestamp, 'yyyy-MM-dd HH:mm')
| order by Timestamp asc;
```

```
// Calculate summary statistics
let SummaryStats = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp between (IncidentStart .. IncidentEnd)
| summarize
    TotalEmails = count(),
    TotalAutoReplies = countif(Subject contains_any ("out of office", "automatic reply")),
    TotalMalicious = countif(ThreatTypes != ""),
    UniqueSenders = dcount(SenderFromAddress),
    UniqueDomains = dcount(SenderFromDomain),
    PeakHour = topk(count(), 1, bin(Timestamp, 1h)),
    DurationHours = datetime_diff('hour', max(Timestamp), min(Timestamp))
| extend
    AutoReplyRate = round((TotalAutoReplies * 100.0) / TotalEmails, 2),
    ThreatRate = round((TotalMalicious * 100.0) / TotalEmails, 2),
    AvgEmailsPerHour = round(TotalEmails * 1.0 / DurationHours, 0);

// Generate executive summary
let ExecutiveSummary = SummaryStats
| extend
    IncidentClassification = case(
        ThreatRate > 5, "TRUE POSITIVE - Malicious Mail Bombing Attack",
        AutoReplyRate > 70 and ThreatRate < 1, "FALSE POSITIVE - Legitimate Auto-Reply Surge",
        "INDETERMINATE - Further Investigation Required"
    ),
    SeverityAssessment = case(
        ThreatRate > 10, "CRITICAL",
        ThreatRate > 5, "HIGH",
        AvgEmailsPerHour > 500, "MEDIUM",
        "LOW"
    ),
    RecommendedAction = case(
        ThreatRate > 5, "IMMEDIATE: Block senders, isolate mailbox, forensic analysis",
```

```
        AutoReplyRate > 70, "Close as false positive, tune detection rules",
        "Continue investigation, verify with mailbox owner"
    )
| project
    IncidentMailbox = IncidentMailbox,
    IncidentStart,
    IncidentEnd,
    DurationHours,
    TotalEmails,
    AvgEmailsPerHour,
    TotalAutoReplies,
    AutoReplyRate,
    TotalMalicious,
    ThreatRate,
    UniqueSenders,
    UniqueDomains,
    IncidentClassification,
    SeverityAssessment,
    RecommendedAction;

// Display complete timeline with summary
print "╔═══════════════════════════════════════════════════════════════════════════════╗"
| union (print "║          INCIDENT #30801 - TIMELINE ANALYSIS                  ║")
| union (print "╚═══════════════════════════════════════════════════════════════════════════════╝")
| union (print "")
| union (print "=== EXECUTIVE SUMMARY ===")
| union (ExecutiveSummary)
| union (print "")
| union (print "=== DETAILED TIMELINE ===")
| union (TimelineEvents)
| union (print "")
| union (print "=== VISUALIZATION ===")
| union (
```

```
    TimelineEvents
    | render timechart with (
        title="Incident Timeline by Event Type",
        xtitle="Time",
        ytitle="Email Count"
    )
)
```

Query 9: Investigation Summary Dashboard

kql

```
// ============================================================
// QUERY 9: MASTER INVESTIGATION DASHBOARD
// Purpose: Single query for complete incident analysis
// Use Case: Quick assessment for incident response team
// ============================================================

let IncidentMailbox = "info@borealisgroup.com";
let AnalysisWindow = 48h;

// Volume metrics
let VolumeMetrics = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(AnalysisWindow)
| summarize
    TotalVolume = count(),
    PeakHourVolume = max(count()) by bin(Timestamp, 1h),
    AvgHourlyVolume = avg(count()) by bin(Timestamp, 1h)
| summarize
    Total = sum(TotalVolume),
    Peak = max(PeakHourVolume),
    Average = round(avg(AvgHourlyVolume), 0);

// Content analysis
let ContentMetrics = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(AnalysisWindow)
| extend IsAutoReply = iff(
    Subject contains_any ("out of office", "automatic reply", "away"), 1, 0
)
| summarize
    AutoReplies = sum(IsAutoReply),
    TotalEmails = count(),
    UniqueSenders = dcount(SenderFromAddress),
```

```
    UniqueDomains = dcount(SenderFromDomain)
| extend
    AutoReplyPercent = round(AutoReplies * 100.0 / TotalEmails, 2),
    EmailsPerSender = round(TotalEmails * 1.0 / UniqueSenders, 2);

// Threat assessment
let ThreatMetrics = EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(AnalysisWindow)
| join kind=leftouter (
    EmailUrlInfo | where Timestamp > ago(AnalysisWindow)
) on NetworkMessageId
| join kind=leftouter (
    EmailAttachmentInfo | where Timestamp > ago(AnalysisWindow)
) on NetworkMessageId
| summarize
    MaliciousEmails = dcountif(NetworkMessageId, ThreatTypes != ""),
    TotalEmails = dcount(NetworkMessageId),
    EmailsWithURLs = dcountif(NetworkMessageId, isnotempty(Url)),
    EmailsWithAttachments = dcountif(NetworkMessageId, isnotempty(FileName))
| extend
    ThreatPercentage = round(MaliciousEmails * 100.0 / TotalEmails, 2),
    URLRate = round(EmailsWithURLs * 100.0 / TotalEmails, 2),
    AttachmentRate = round(EmailsWithAttachments * 100.0 / TotalEmails, 2);

// Campaign correlation
let CampaignCheck = EmailEvents
| where SenderFromAddress has "borealisgroup.com"
| where Timestamp > ago(AnalysisWindow + 24h) and Timestamp < ago(AnalysisWindow)
| where Subject has_any ("fraud", "alert", "security", "warning")
| summarize
    CampaignEmails = count(),
    CampaignRecipients = dcount(RecipientEmailAddress)
```

```
| extend CampaignDetected = iff(CampaignEmails > 100, "✅ Yes", "❌ No");

// Final assessment
print "╔═══════════════════════════════════════════════════════════════════════════════╗"
| union (print "║        MAIL BOMBING INVESTIGATION - EXECUTIVE DASHBOARD               ║")
| union (print "╚═══════════════════════════════════════════════════════════════════════════════╝")
| union (print "")
| union (print "📊 VOLUME METRICS:")
| union (
    VolumeMetrics
    | extend Summary = strcat(
        "   • Total Emails: ", Total, "\n",
        "   • Peak Hour Volume: ", Peak, "\n",
        "   • Average per Hour: ", Average
    )
    | project Summary
)
| union (print "")
| union (print "📧 CONTENT ANALYSIS:")
| union (
    ContentMetrics
    | extend Summary = strcat(
        "   • Auto-Reply Rate: ", AutoReplyPercent, "%\n",
        "   • Unique Senders: ", UniqueSenders, "\n",
        "   • Unique Domains: ", UniqueDomains, "\n",
        "   • Emails per Sender: ", EmailsPerSender
    )
    | project Summary
)
| union (print "")
| union (print "🛡️ THREAT ASSESSMENT:")
| union (
    ThreatMetrics
```

```
    | extend Summary = strcat(
        "   • Malicious Content: ", ThreatPercentage, "%\n",
        "   • Emails with URLs: ", URLRate, "%\n",
        "   • Emails with Attachments: ", AttachmentRate, "%"
    )
    | project Summary
)
| union (print "")
| union (print "🔗 CAMPAIGN CORRELATION:")
| union (
    CampaignCheck
    | extend Summary = strcat(
        "   • Outbound Campaign Detected: ", CampaignDetected, "\n",
        "   • Campaign Size: ", CampaignEmails, " emails\n",
        "   • Recipients: ", CampaignRecipients
    )
    | project Summary
)
| union (print "")
| union (print "🎯 CONCLUSION:")
| union (
    VolumeMetrics
    | join (ContentMetrics) on $left.Total == $right.TotalEmails
    | join (ThreatMetrics) on $left.Total == $right.TotalEmails1
    | extend
        Conclusion = case(
            ThreatPercentage > 5, "🔴 TRUE POSITIVE - Malicious mail bombing attack detected",
            AutoReplyPercent > 70 and ThreatPercentage < 1, "🟢 FALSE POSITIVE - Legitimate auto-reply surge",
            "🟡 REQUIRES INVESTIGATION - Mixed indicators"
        ),
        Severity = case(
            ThreatPercentage > 10, "CRITICAL",
            ThreatPercentage > 5, "HIGH",
```

```
        AutoReplyPercent > 70, "LOW - False Alarm",
        "MEDIUM"
    ),
    RecommendedAction = case(
        ThreatPercentage > 5, "IMMEDIATE: Block senders, isolate mailbox, forensic investigation",
        AutoReplyPercent > 70, "Close as false positive, implement rule tuning per Solution 1",
        "Continue monitoring, coordinate with mailbox owner"
    )
| project
    Assessment = strcat(
        "  Classification: ", Conclusion, "\n",
        "  Severity: ", Severity, "\n",
        "  Action: ", RecommendedAction
    )
)
```

Query 10: Export Report for Documentation

kql

```
// ============================================================
// QUERY 10: FORMATTED INCIDENT REPORT EXPORT
// Purpose: Generate formatted report for ServiceNow/ticketing
// Use Case: Incident documentation and closure
// ============================================================

let IncidentNumber = "30801";
let IncidentMailbox = "info@borealisgroup.com";
let AnalysisWindow = 48h;
let AnalystName = "Your Name";  // Update this

EmailEvents
| where RecipientEmailAddress == IncidentMailbox
| where Timestamp > ago(AnalysisWindow)
| extend
    IsAutoReply = iff(Subject contains_any ("out of office", "automatic reply"), "Yes", "No"),
    IsMalicious = iff(ThreatTypes != "", "Yes", "No")
| summarize
    TotalEmails = count(),
    AutoReplies = countif(IsAutoReply == "Yes"),
    MaliciousEmails = countif(IsMalicious == "Yes"),
    UniqueSenders = dcount(SenderFromAddress),
    UniqueDomains = dcount(SenderFromDomain),
    FirstEmail = min(Timestamp),
    LastEmail = max(Timestamp),
    TopSenders = make_set(SenderFromAddress, 10),
    TopDomains = make_set(SenderFromDomain, 10)
| extend
    AutoReplyRate = round((AutoReplies * 100.0) / TotalEmails, 2),
    ThreatRate = round((MaliciousEmails * 100.0) / TotalEmails, 2),
    DurationHours = datetime_diff('hour', LastEmail, FirstEmail),
    Classification = case(
        ThreatRate > 5, "True Positive - Malicious Attack",
```

```
        AutoReplyRate > 70, "False Positive - Legitimate Auto-Replies",
        "Indeterminate"
    ),
    Severity = case(
        ThreatRate > 5, "High",
        AutoReplyRate > 70, "Low",
        "Medium"
    )
| project
    ReportTitle = "Mail Bombing Incident Analysis Report",
    IncidentID = IncidentNumber,
    AffectedAsset = IncidentMailbox,
    AnalysisDate = now(),
    Analyst = AnalystName,
    blank1 = "",
    SectionVolume = "--- VOLUME ANALYSIS ---",
    TotalEmailVolume = TotalEmails,
    IncidentDuration = strcat(DurationHours, " hours"),
    FirstDetected = FirstEmail,
    LastDetected = LastEmail,
    blank2 = "",
    SectionContent = "--- CONTENT BREAKDOWN ---",
    AutoReplyCount = AutoReplies,
    AutoReplyPercentage = strcat(AutoReplyRate, "%"),
    MaliciousEmailCount = MaliciousEmails,
    ThreatPercentage = strcat(ThreatRate, "%"),
    blank3 = "",
    SectionSources = "--- SOURCE ANALYSIS ---",
    UniqueSenderCount = UniqueSenders,
    UniqueDomainCount = UniqueDomains,
    TopSendingDomains = strcat_array(TopDomains, ", "),
    blank4 = "",
    SectionConclusion = "--- CONCLUSION ---",
```

```
IncidentClassification = Classification,
AssignedSeverity = Severity,
RecommendedDisposition = case(
    Classification contains "False Positive", "Close as Benign Positive - Tune detection rules",
    Classification contains "True Positive", "Escalate to Tier 2 - Full investigation required",
    "Continue investigation"
),
blank5 = "",
SectionActions = "--- RECOMMENDED ACTIONS ---",
Action1 = "✓ Document findings in incident ticket",
Action2 = case(
    Classification contains "False Positive",
    "✓ Implement Solution 1 (Detection Rule Optimization)",
    "✓ Block malicious senders and isolate mailbox"
),
Action3 = case(
    Classification contains "False Positive",
    "✓ Communicate with client per template",
    "✓ Conduct forensic analysis of malicious emails"
),
Action4 = "✓ Update incident response playbook",
blank6 = "",
ReportGenerated = format_datetime(now(), 'yyyy-MM-dd HH:mm:ss UTC')
```

📝 IMPLEMENTATION CHECKLIST

Phase 1: Immediate Actions (Day 1)

- [ ] Run Query 1-4 to complete initial investigation
- [ ] Execute Query 9 for executive summary
- [ ] Document findings using Query 10
- [ ] Close Incident #30801 with proper classification

- [ ] Communicate with client using provided template

Phase 2: Short-term (Week 1-2)

- [ ] Implement Solution 2 (Mail Flow Filtering)
- [ ] Test filtering rules in pilot environment
- [ ] Deploy to production after validation
- [ ] Monitor for 1 week to ensure effectiveness

Phase 3: Long-term (Month 1-2)

- [ ] Implement Solution 1 (Detection Rule Optimization)
- [ ] Create/update VIP mailbox watchlist
- [ ] Deploy enhanced detection rule
- [ ] Conduct 2-week testing phase
- [ ] Measure and document improvement metrics

Phase 4: Continuous Improvement

- [ ] Monthly false positive rate review
- [ ] Quarterly threshold optimization
- [ ] Update playbooks based on lessons learned
- [ ] Share findings with SOC team

🔗 QUICK REFERENCE

Investigation Flow

1. Volume Analysis (Query 1) → Understand scale
2. Auto-Reply Check (Query 2) → Identify pattern
3. Threat Assessment (Query 4) → Confirm safety
4. Campaign Correlation (Query 5) → Find root cause

5. Dashboard Review (Query 9) → Make decision
6. Document & Close (Query 10) → Complete incident

Decision Matrix

Auto-Reply %	Threat %	Classification	Action
> 70%	< 1%	False Positive	Close + Tune Rules
30-70%	< 1%	Indeterminate	Investigate Further
Any	> 5%	True Positive	Escalate + Block
< 30%	1-5%	Suspicious	Deep Investigation

Key Thresholds

- **Generic Mailboxes:** 1500 emails/hour = Investigation trigger
- **User Mailboxes:** 200 emails/hour = Investigation trigger
- **VIP Mailboxes:** 100 emails/hour = Investigation trigger
- **Auto-Reply Rate:** > 70% = Likely false positive
- **Threat Rate:** > 5% = Confirmed threat

📚 ADDITIONAL RESOURCES

Microsoft Documentation

- Microsoft Defender for Office 365
- Mail flow rules
- Anti-spam policies

KQL Resources

- KQL Quick Reference
- EmailEvents Schema
- Sentinel Analytics Rules

Document Version: 1.0
Last Updated: November 3, 2025
Next Review Date: February 3, 2026
Classification: Internal Use - SOC Team

📞 SUPPORT CONTACTS

For questions about this document:

- SOC Manager: [Contact Info]
- Security Engineering: [Contact Info]
- Microsoft Support: Premier Support Portal

For incident escalation:

- Tier 2 SOC: [Contact Info]
- Incident Commander: [Contact Info]
- Client CISO: [Contact Info]

End of Document