Scattered Spider Threat Hunting Runbook

1) Purpose

This threat hunting runbook provides Microsoft Sentinel KQL queries and investigative procedures to detect and hunt for Scattered Spider (UNC3944, Starfraud, Scatter Swine, Muddled Libra) threat actor activities. The runbook focuses on their signature tactics including social engineering, MFA fatigue attacks, credential dumping, and cloud service abuse for data exfiltration.

2) Threat Context

Actor Information

- **Primary Name**: Scattered Spider
- **Aliases**: UNC3944, Starfraud, Scatter Swine, Muddled Libra
- **Composition**: Primarily young adults and teenagers
- **Geographic Focus**: English-speaking countries (US, UK, Canada, Australia)

Motivation

- Financial extortion through ransomware deployment
- Data theft for sale or leverage
- Double extortion combining data theft with encryption

Key TTPs

- **T1566.004**: Vishing (Voice Phishing) - Primary initial access vector
- **T1621**: MFA Fatigue attacks
- **T1078.004**: Valid cloud account abuse
- **T1003.003**: NTDS credential dumping
- **T1567.002**: Exfiltration to cloud storage (MEGA.NZ)
- **T1486**: Ransomware deployment (BlackCat/ALPHV, DragonForce)
- **T1055**: Process Injection (Spectre RAT - 2025)

- **T1027**: Obfuscated Files or Information (Spectre RAT - 2025)
- **T1053**: Scheduled Task/Job (Spectre RAT - 2025)

3) Technical Prerequisites for Threat Hunting

Required Data Sources

- Microsoft Defender for Endpoint (MDE)
- Azure Active Directory Sign-in logs
- Office 365 audit logs
- Microsoft Defender for Cloud Apps
- Windows Security Event logs (4624, 4625, 4648, 4776)
- Sysmon (Events 1, 3, 7, 10, 11)
- Network connection logs
- VPN authentication logs

Recommended Log Retention

- Minimum 90 days for correlation analysis
- 180 days recommended for campaign tracking

4) Threat Hunting Hypotheses

Hypothesis 1: Voice Phishing (Vishing) Attacks Detection

Mapping: T1566.004 - Spearphishing via Voice **Hypothesis Explanation**: Scattered Spider uses vishing to manipulate help desk personnel and employees into providing credentials or installing remote access tools. **Hunting Focus**: Detect unusual help desk ticket patterns, after-hours password resets, and rapid remote tool installations.

```
// Hunt for suspicious password reset patterns following help desk
contacts
let timeRange = 7d;
let suspiciousResets =
    SigninLogs
    | where TimeGenerated > ago(timeRange)
    | where ResultType == 0
    | where AuthenticationDetails has_any ("ResetPassword")
    | summarize ResetCount = count(),
                UniqueLocations = dcount(Location),
                UniqueIPs = dcount(IPAddress),
                FirstReset = min(TimeGenerated),
                LastReset = max(TimeGenerated)
                by UserPrincipalName
    | where ResetCount >= 3 and UniqueLocations >= 2
    | extend SuspiciousScore = ResetCount * UniqueLocations;
// Correlate with remote access tool installations
let remoteTools =
    DeviceProcessEvents
    | where TimeGenerated > ago(timeRange)
```

```
    | where ProcessCommandLine has_any ("teamviewer", "anydesk",
"connectwise", "screenconnect", "splashtop")
    | project DeviceName, ProcessCommandLine, TimeGenerated, AccountName;
suspiciousResets
| join kind=inner (remoteTools) on $left.UserPrincipalName ==
$right.AccountName
| where datetime_diff('minute', LastReset, TimeGenerated) between (0 ..
120)
| project UserPrincipalName, SuspiciousScore, ResetCount, UniqueLocations,
RemoteTool = ProcessCommandLine
```

Your organisation might run selfservice password resets. However, not all the reset requests are captured by SigningLogs, so there is a much more verbouse way to investigate over password resets. The next query will give the top ten password changers of the last 7d, and it's worth to check if you spot admin accounts in there.

```
let timeInterval = 7d;
let signInResets =
SigninLogs
| where TimeGenerated > ago(timeInterval)
| where AuthenticationRequirement has "ResetPassword"
   or AuthenticationDetails has "ResetPassword"
| project TimeGenerated, User = UserPrincipalName;

let auditResets =
AuditLogs
| where TimeGenerated > ago(timeInterval)
| where ActivityDisplayName has "Reset user password" or
ActivityDisplayName has "Self-service password reset"
| extend User = tostring(TargetResources[0].userPrincipalName)
| project TimeGenerated, User;

union signInResets, auditResets
| summarize ResetCount = count() by User
| sort by ResetCount desc
```

Investigation Steps:

1. Review user's recent authentication patterns and geographic anomalies
2. Check for help desk tickets or support requests within 2 hours of resets
3. Examine installed remote access tools and their usage patterns
4. Validate with user if password reset was legitimate
5. Validate if you spot suspicious and too high recurring resets for users, with a special eye to admin accounts

Hypothesis 2: MFA Fatigue Attack Detection

Mapping: T1621 - Multi-Factor Authentication Request Generation **Hypothesis Explanation**: Attackers bombard users with MFA push notifications to overwhelm them into approving fraudulent requests.

Hunting Focus: Identify users receiving excessive MFA prompts followed by successful authentication.

```
// Detect MFA fatigue patterns
let timeWindow = 1h;
let mfaEvents =
    SigninLogs
    | where TimeGenerated > ago(7d)
    | where AuthenticationRequirement == "multiFactorAuthentication"
    | where ResultType in (50074, 50076, 0) // Failed MFA, MFA required,
Success
    | extend MFAResult = case(
        ResultType == 0, "Success",
        ResultType == 50074, "MFA_Challenge_Declined",
        ResultType == 50076, "MFA_Challenge_Timeout",
        "Other"
    );
// Find users with high MFA challenge volume followed by success
mfaEvents
| summarize
    TotalChallenges = countif(MFAResult != "Success"),
    SuccessfulAuths = countif(MFAResult == "Success"),
    FirstChallenge = min(TimeGenerated),
    LastSuccess = max(TimeGenerated),
    UniqueIPs = dcount(IPAddress),
    Locations = make_set(Location)
    by UserPrincipalName, bin(TimeGenerated, timeWindow)
| where TotalChallenges >= 5 and SuccessfulAuths >= 1
| where datetime_diff('minute', LastSuccess, FirstChallenge) <= 60
| extend FatigueScore = TotalChallenges * UniqueIPs
| order by FatigueScore desc
```

Investigation Steps:

1. Contact user to verify if they initiated the authentication attempts
2. Review source IP addresses and geographic locations for anomalies
3. Check for subsequent suspicious activities from successful sessions
4. Examine device compliance and registration status

Hypothesis 3: Credential Dumping via NTDS

Mapping: T1003.003 - OS Credential Dumping: NTDS **Hypothesis Explanation**: Scattered Spider dumps Active Directory credentials using tools like Mimikatz to escalate privileges. **Hunting Focus**: Detect NTDS.dit access, credential dumping tools, and related artifacts.

```
// Hunt for NTDS credential dumping activities
let credentialDumpingTools = dynamic(["mimikatz", "procdump",
"comsvcs.dll", "ntdsutil"]);
let ntdsAccess =
    DeviceFileEvents
    | where TimeGenerated > ago(7d)
```

```
    | where FileName =~ "ntds.dit" or FolderPath has "ntds"
    | where ActionType in ("FileCreated", "FileModified", "FileCopied")
    | project TimeGenerated, DeviceName, FileName, FolderPath,
InitiatingProcessCommandLine, InitiatingProcessAccountName;
let suspiciousProcesses =
    DeviceProcessEvents
    | where TimeGenerated > ago(7d)
    | where ProcessCommandLine has_any (credentialDumpingTools)
    | project TimeGenerated, DeviceName, ProcessCommandLine, AccountName,
InitiatingProcessCommandLine;
// Combine NTDS access with credential dumping tools
ntdsAccess
| union suspiciousProcesses
| summarize Events = make_list(pack_all()),
            EventTypes = make_set(strcat(FileName, ProcessCommandLine)),
            FirstSeen = min(TimeGenerated),
            LastSeen = max(TimeGenerated)
            by DeviceName, AccountName
| where array_length(Events) >= 2
| extend RiskScore = array_length(EventTypes) * 10
```

Investigation Steps:

1. Verify if account has legitimate administrative privileges
2. Check for recent privilege escalation activities
3. Review related process execution and parent processes
4. Examine network connections from affected device
5. Look for lateral movement indicators

Hypothesis 4: Cloud Storage Exfiltration to MEGA.NZ

Mapping: T1567.002 - Exfiltration to Cloud Storage **Hypothesis Explanation**: Scattered Spider primarily uses MEGA.NZ for data exfiltration due to its encryption capabilities. **Hunting Focus**: Detect large data uploads to MEGA.NZ and other cloud storage services.

```
// Hunt for data exfiltration to cloud storage services
let cloudStorageServices = dynamic(["mega.nz", "mega.co.nz",
"dropbox.com", "onedrive.live.com", "drive.google.com"]);
let largeUploads =
    DeviceNetworkEvents
    | where TimeGenerated > ago(7d)
    | where RemoteUrl has_any (cloudStorageServices)
    | where LocalPort != 443 or RemotePort != 443
    | summarize
        TotalConnections = count(),
        FirstConnection = min(TimeGenerated),
        LastConnection = max(TimeGenerated),
        UniqueDestinations = dcount(RemoteUrl)
        by DeviceName, InitiatingProcessAccountName, RemoteUrl
    | where TotalConnections >= 10;
```

```
// Correlate with file compression activities
let compressionActivity =
    DeviceProcessEvents
    | where TimeGenerated > ago(7d)
    | where ProcessCommandLine has_any ("zip", "rar", "7z", "tar", "gzip")
    | where ProcessCommandLine contains "-r" or ProcessCommandLine
contains "archive"
    | project TimeGenerated, DeviceName, InitiatingProcessAccountName,
ProcessCommandLine;

largeUploads
| join kind=leftouter (compressionActivity) on DeviceName,
InitiatingProcessAccountName
| where datetime_diff('hour', FirstConnection, TimeGenerated) between (-2
.. 2)
| project
    DeviceName,
    AccountName = InitiatingProcessAccountName,
    RemoteUrl,
    TotalConnections,
    FirstConnection,
    CompressionCommand = ProcessCommandLine
```

Investigation Steps:

1. Identify compressed file locations and contents
2. Determine if data contains sensitive information
3. Review user's normal cloud storage usage patterns
4. Check for data classification violations
5. Examine concurrent credential access attempts

Hypothesis 5: Ransomware Deployment Detection

Mapping: T1486 - Data Encrypted for Impact **Hypothesis Explanation**: Scattered Spider deploys BlackCat/ALPHV ransomware as final impact stage. **Hunting Focus**: Detect rapid file encryption patterns and ransomware indicators.

```
// Hunt for rapid file encryption and ransomware deployment
let encryptionExtensions = dynamic([".blackcat", ".alphv", ".encrypted",
".locked", ".crypto"]);
let rapidEncryption =
    DeviceFileEvents
    | where TimeGenerated > ago(1d)
    | where ActionType == "FileModified"
    | where FileName has_any (encryptionExtensions)
        or (FileName contains "readme" and FileName contains "txt")
        or (FileName contains "ransom" and FileName contains "note")
    | extend FileExtension = extract(@"\.([a-zA-Z0-9]+)$", 1, FileName)
    | summarize
        FilesEncrypted = count(),
        FirstEncryption = min(TimeGenerated),
```

```
        LastEncryption = max(TimeGenerated),
        UniqueDirectories = dcount(FolderPath),
        Extensions = make_set(FileExtension)
        by DeviceName, InitiatingProcessAccountName,
InitiatingProcessCommandLine
    | where FilesEncrypted >= 50 and UniqueDirectories >= 5;

// Look for ransomware process indicators
let ransomwareProcesses =
    DeviceProcessEvents
    | where TimeGenerated > ago(1d)
    | where ProcessCommandLine has_any ("blackcat", "alphv", "encrypt",
"ransom")
    | where ProcessCommandLine contains ".exe" or ProcessCommandLine
contains "powershell"
    | project
        DeviceName,
        InitiatingProcessAccountName,
        ProcessCommandLine,
        TimeGenerated;

// Combine and flag ransomware indicators
rapidEncryption
| project DeviceName, InitiatingProcessAccountName, FilesEncrypted,
FirstEncryption
| union (
    ransomwareProcesses
    | extend FilesEncrypted = 0, FirstEncryption = TimeGenerated
    | project DeviceName, InitiatingProcessAccountName, FilesEncrypted,
FirstEncryption
)
| summarize
    TotalEncryptionEvents = sum(FilesEncrypted),
    FirstObserved = min(FirstEncryption),
    LastObserved = max(FirstEncryption)
    by DeviceName, InitiatingProcessAccountName
| extend CriticalAlert = "RANSOMWARE_DETECTED"
```

Investigation Steps: Obviously this query will most likely, and hopefully, return nothing.

1. Immediately isolate affected systems
2. Identify ransomware variant and encryption scope
3. Check for ransom notes and contact information
4. Review backup integrity and recovery options
5. Analyze attack timeline and entry vector

Hypothesis 6: 2025 Domain Pattern Detection (Technology Vendor Impersonation)

Mapping: T1566.002 - Phishing via Service (2025 Evolution) **Hypothesis Explanation**: Scattered Spider has evolved to use subdomain-based domain patterns impersonating technology vendors (SSO, IdP, VPN

providers). **Hunting Focus**: Detect new 2025 domain patterns focusing on technology vendor impersonation.

```
// Hunt for 2025 evolved domain patterns
let suspiciousDomains = dynamic([
    "sso.", "okta.", "vpn.", "helpdesk.", "support.",
    "admin.", "portal.", "login.", "auth."
]);
DnsEvents
| where TimeGenerated > ago(30d)
| where Name has_any (suspiciousDomains)
| where not(Name has_any ("microsoft.com", "okta.com", "google.com"))
| summarize DNSRequests = count(),
            FirstSeen = min(TimeGenerated),
            LastSeen = max(TimeGenerated)
            by ClientIP, Name
| where DNSRequests >= 3
| extend RiskScore = DNSRequests * 2
| order by RiskScore desc
```

Investigation Steps:

1. Verify if domain is legitimate technology vendor infrastructure
2. Check domain registration details and creation date
3. Review associated network traffic and user interactions
4. Correlate with authentication events for potential credential compromise

Hypothesis 7: Spectre RAT Detection (2025 New Malware)

Mapping: T1055, T1027, T1053 - Process Injection, Obfuscation, Scheduled Tasks **Hypothesis Explanation**: Scattered Spider has deployed new Spectre RAT with advanced obfuscation and evasion capabilities. **Hunting Focus**: Detect Spectre RAT indicators including process injection, obfuscation, and persistent scheduled tasks.

NOTE: THIS QUERY IS PRONE TO A HUGE NUMBER OF FALSE POSITIVES, use with caution

```
// Hunt for Spectre RAT indicators
let spectreIndicators = dynamic([
    "spectre", "crypter", "obfusc", "inject"
]);
DeviceProcessEvents
| where TimeGenerated > ago(7d)
| where ProcessCommandLine has_any (spectreIndicators) or
        FileName has_any (spectreIndicators)
| project TimeGenerated, DeviceName, FileName, ProcessCommandLine,
          AccountName, InitiatingProcessAccountName
| join kind=leftouter (
    DeviceNetworkEvents
    | where TimeGenerated > ago(7d)
```

```
    | project DeviceName, RemoteIP, RemotePort, InitiatingProcessFileName
    ) on DeviceName
| where isnotempty(RemoteIP)
| extend ThreatScore = 85
```

Investigation Steps:

1. Analyze process injection techniques and target processes
2. Review network communications to identify C2 infrastructure
3. Check for scheduled task creation and persistence mechanisms
4. Examine file obfuscation methods and crypter signatures

5) Summary of Runbook

Hunt Hypothesis	MITRE TTP	KQL Query Focus	Detection Priority
Vishing Detection	T1566.004	Password resets + Remote tools	Critical
MFA Fatigue	T1621	Excessive MFA challenges	Critical
NTDS Dumping	T1003.003	Credential dumping tools	High
Cloud Exfiltration	T1567.002	MEGA.NZ uploads	Critical
Ransomware Deployment	T1486	File encryption patterns	Critical
2025 Domain Patterns	T1566.002	Technology vendor impersonation	**Critical**
Spectre RAT (2025)	T1055, T1027, T1053	Process injection + Obfuscation	**High**

Key Detection Metrics

- **Coverage**: 8 critical TTPs including 2025 evolutions with high-confidence detection logic
- **False Positive Rate**: Low due to correlation-based queries and behavioral analysis
- **Response Time**: Automated alerting for critical findings
- **Investigation Depth**: Multi-stage verification procedures
- **2025 Enhancements**: Updated for technology vendor impersonation, Spectre RAT, and MSP targeting

6) References

- https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-320a
- https://attack.mitre.org/groups/G1015/
- https://www.microsoft.com/security/blog/2023/09/14/hunt-for-scattered-spider-indicators

This document is prepared by Crimson7 - 2025 v1.1 (Updated with 2025 Intelligence)