UAC Bypass via Windows Firewall Snap-In Hijack

Identifies attempts to bypass User Account Control (UAC) by hijacking the Microsoft Management Console (MMC) Windows Firewall snap-in. Attackers bypass UAC to stealthily execute code with elevated permissions.

Rule type: eql

Rule indices:

- winlogbeat-*
- logs-endpoint.events.process-*
- logs-windows.sysmon_operational-*
- endgame-*
- logs-m365_defender.event-*
- logs-sentinel_one_cloud_funnel.*

Severity: medium

Risk score: 47

Runs every: 5m

Searches indices from: now-9m (Date Math format, see also `Additional look-back time`)

Maximum alerts per execution: 100

References:

- https://github.com/AzAgarampur/byeintegrity-uac

Tags:

- Domain: Endpoint
- OS: Windows
- Use Case: Threat Detection
- Tactic: Privilege Escalation
- Tactic: Defense Evasion
- Resources: Investigation Guide
- Data Source: Elastic Endgame
- Data Source: Elastic Defend
- Data Source: Sysmon
- Data Source: Microsoft Defender for Endpoint
- Data Source: SentinelOne

Version: 312

Rule authors:

- Elastic

Rule license: Elastic License v2

Investigation guide

edit

Triage and analysis

Investigating UAC Bypass via Windows Firewall Snap-In Hijack

Windows User Account Control (UAC) allows a program to elevate its privileges (tracked as low to high integrity levels) to

perform a task under administrator-level permissions, possibly by prompting the user for confirmation. UAC can deny an operation under high-integrity enforcement, or allow the user to perform the action if they are in the local administrators group and enter an administrator password when prompted.

For more information about the UAC and how it works, check the official Microsoft docs page.

This rule identifies attempts to bypass User Account Control (UAC) by hijacking the Microsoft Management Console (MMC) Windows Firewall snap-in. Attackers bypass UAC to stealthily execute code with elevated permissions.

Note: This investigation guide uses the Osquery Markdown Plugin introduced in Elastic Stack version 8.5.0. Older Elastic Stack versions will display unrendered Markdown in this guide.

Possible investigation steps

- Investigate the process execution chain (parent process tree) for unknown processes. Examine their executable files for prevalence, whether they are located in expected locations, and if they are signed with valid digital signatures.
- Investigate other alerts associated with the user/host during the past 48 hours.
- Inspect the host for suspicious or abnormal behavior in the alert timeframe.
- Investigate any abnormal behavior by the subject process such as network connections, registry or file modifications, and any spawned child processes.
- Examine the host for derived artifacts that indicate suspicious activities:
- Analyze any suspicious spawned processes using a private sandboxed analysis system.

- Observe and collect information about the following activities in both the sandbox and the alert subject host:
- Attempts to contact external domains and addresses.
- Use the Elastic Defend network events to determine domains and addresses contacted by the subject process by filtering by the process' `process.entity_id`.
- Examine the DNS cache for suspicious or anomalous entries.
- !{osquery{"label":"Osquery - Retrieve DNS Cache","query":"SELECT * FROM dns_cache"}}
- Use the Elastic Defend registry events to examine registry keys accessed, modified, or created by the related processes in the process tree.
- Examine the host services for suspicious or anomalous entries.
- !{osquery{"label":"Osquery - Retrieve All Services","query":"SELECT description, display_name, name, path, pid, service_type, start_type, status, user_account FROM services"}}
- !{osquery{"label":"Osquery - Retrieve Services Running on User Accounts","query":"SELECT description, display_name, name, path, pid, service_type, start_type, status, user_account FROM services WHERE\nNOT (user_account LIKE *%LocalSystem* OR user_account LIKE *%LocalService* OR user_account LIKE *%NetworkService* OR\nuser_account == null)\n"}}
- !{osquery{"label":"Osquery - Retrieve Service Unsigned Executables with Virustotal Link","query":"SELECT concat(*https://www.virustotal.com/gui/file/*, sha1) AS VtLink, name, description, start_type, status, pid,\nservices.path FROM services JOIN authenticode ON services.path = authenticode.path OR services.module_path =\nauthenticode.path JOIN hash ON services.path = hash.path WHERE authenticode.result != *trusted*\n"}}

- Retrieve the files' SHA-256 hash values using the PowerShell `Get-FileHash` cmdlet and search for the existence and reputation of the hashes in resources like VirusTotal, Hybrid-Analysis, CISCO Talos, Any.run, etc.
- Investigate potentially compromised accounts. Analysts can do this by searching for login events (for example, 4624) to the target host after the registry modification.

False positive analysis

- This activity is unlikely to happen legitimately. Benign true positives (B-TPs) can be added as exceptions if necessary.

Response and remediation

- Initiate the incident response process based on the outcome of the triage.
- Isolate the involved host to prevent further post-compromise behavior.
- If the triage identified malware, search the environment for additional compromised hosts.
- Implement temporary network rules, procedures, and segmentation to contain the malware.
- Stop suspicious processes.
- Immediately block the identified indicators of compromise (IoCs).
- Inspect the affected systems for additional malware backdoors like reverse shells, reverse proxies, or droppers that attackers could use to reinfect the system.
- Remove and block malicious artifacts identified during triage.
- Run a full antimalware scan. This may reveal additional artifacts left in the system, persistence mechanisms, and malware components.
- Investigate credential exposure on systems compromised or used by the attacker to ensure all compromised accounts are identified. Reset passwords for these

accounts and other potentially compromised credentials, such as email, business systems, and web services.

- Determine the initial vector abused by the attacker and take action to prevent reinfection through the same vector.
- Using the incident response data, update logging and audit policies to improve the mean time to detect (MTTD) and the mean time to respond (MTTR).

Rule query

edit

```
process where host.os.type == "windows" and event.ty
 process.parent.name == "mmc.exe" and
 /* process.Ext.token.integrity_level_name == "high"
 /* args of the Windows Firewall SnapIn */
  process.parent.args == "WF.msc" and process.name !
```

Framework: MITRE ATT&CK™

- Tactic:
  - Name: Privilege Escalation
  - ID: TA0004
  - Reference URL: https://attack.mitre.org/tactics/TA0004/
- Technique:
  - Name: Abuse Elevation Control Mechanism
  - ID: T1548
  - Reference URL: https://attack.mitre.org/techniques/T1548/
- Sub-technique:
  - Name: Bypass User Account Control

    - ID: T1548.002
    - Reference URL: https://attack.mitre.org/techniques/T1548/002/
- Tactic:
    - Name: Defense Evasion
    - ID: TA0005
    - Reference URL: https://attack.mitre.org/tactics/TA0005/
- Technique:
    - Name: System Binary Proxy Execution
    - ID: T1218
    - Reference URL: https://attack.mitre.org/techniques/T1218/
- Sub-technique:
    - Name: MMC
    - ID: T1218.014
    - Reference URL: https://attack.mitre.org/techniques/T1218/014/
- Technique:
    - Name: Abuse Elevation Control Mechanism
    - ID: T1548
    - Reference URL: https://attack.mitre.org/techniques/T1548/
- Sub-technique:
    - Name: Bypass User Account Control
    - ID: T1548.002
    - Reference URL: https://attack.mitre.org/techniques/T1548/002/

« UAC Bypass via ICMLuaUtil Elevated COM Interface

UID Elevation from Previously Unknown Executable »