Ransomware Incident Response Documentation

Aligned with NIST SP 800-61

Note

This documentation is based entirely on the forensic evidence extracted from the memory analysis reports (REPORT_BEFORE_... and REPORT_AFTER_...). All conclusions are supported by the RAM acquisition performed with DumpIt.exe and analyzed using Volatility 3 on Ubuntu.

1. Incident Summary

A simulated ransomware attack was executed on an isolated Windows 7 SP1 virtual machine using a malicious PowerShell script launched through a command prompt session. The attack modified bait files and created a ransom note on the desktop. A full memory dump was acquired using DumpIt.exe and analyzed using Volatility 3 to determine the attack sequence and artifacts.

2. Preparation (NIST Phase 1)

This phase covers all readiness steps taken before the incident.

2.1 Isolated Laboratory Environment

A dedicated and fully isolated virtual environment was configured using Internal Network mode to ensure safe execution and observation of the attack without affecting external systems.

2.2 Establishing a Baseline

A clean memory snapshot (BEFORE_RANSOM.mem) was captured prior to initiating the attack. Several baseline forensic reports were generated, including:

- Process list (PSLIST BEFORE)
- Command-line arguments (CMDLINE BEFORE)
- Registry hive mappings (HIVELIST BEFORE)
- System activity log (TIMELINE BEFORE)

These were used for comparison with post-incident data.

2.3 Tool Preparation

The forensic workstation (Ubuntu) was equipped with:

- Volatility 3 Framework
- Required plugins for registry and process enumeration
- DumpIt.exe for RAM acquisition on the Windows VM

This ensured rapid evidence collection during the incident.

3. Detection and Analysis (NIST Phase 2)

This phase describes how the incident was identified and analyzed.

3.1 Indicators of Compromise

- Changes to bait files and appearance of encrypted content
- A ransom note file named "RANSOM_NOTE.txt" on the desktop
- Presence of an active cmd.exe process during the incident
- Differences between the BEFORE and AFTER process and registry reports

3.2 Memory Forensic Findings

3.2.1 Attack Execution via cmd.exe

The memory analysis identified the malicious process as:

```
2792    cmd.exe    "C:\Windows\System32\cmd.exe"
```

This confirms that the ransomware script was launched directly from a command prompt.
(Source: AFTER_PSLIST)

3.2.2 Fast-Exit PowerShell Execution

Analysis of command-line data revealed no presence of powershell.exe in memory:

```
(No entry for powershell.exe)
```

This strongly indicates the use of a fast-exit technique where PowerShell executes the payload and terminates immediately, leaving minimal artifacts.
(Source: AFTER_CMDLINE)

3.2.3 Memory Acquisition Confirmation (DumpIt.exe)

The presence of DumpIt.exe in memory validates that the RAM was collected promptly after the attack:

```
2564    DumpIt.exe    "C:\Tools\Comae-Toolkit-
v20230117\x64\DumpIt.exe"
```

(Source: AFTER_PSLIST)

3.2.4 Registry Hive Comparison Shows No Persistence

A comparison of registry hives before and after the attack indicates no new persistence mechanisms, no startup keys, and no modified registry entries.

Before (BEFORE_HIVELIST):

```
\??\C:\Users\Abo-Ali\ntuser.dat
\REGISTRY\MACHINE\SYSTEM
```

After (AFTER_HIVELIST):

```
\??\C:\Users\Abo-Ali\ntuser.dat
\REGISTRY\MACHINE\SYSTEM
```

The identical output confirms that the ransomware simulation did not attempt to establish persistence mechanisms.
(Sources: BEFORE_HIVELIST and AFTER_HIVELIST)

3.2.5 Ransom Note Located in Memory

A full copy of the ransom note text was found inside the post-incident memory dump using string extraction, confirming that the ransomware successfully executed and left user-facing artifacts in RAM.

4. Containment, Eradication, and Recovery (NIST Phase 3)

4.1 Containment

- A full memory dump (AFTER_RANSOM.mem) was acquired immediately after detecting the attack.

- The virtual machine was powered off to prevent further changes to volatile data.
- Memory files and associated reports were transferred to the forensic workstation for offline analysis.

4.2 Eradication

Root Cause

The root cause was identified as a malicious PowerShell script launched from cmd.exe.

Removal

Instead of applying manual cleanup, the system was restored to a clean snapshot. This approach ensures complete removal of malicious artifacts, registry modifications, encrypted files, and potential persistence mechanisms.

4.3 Recovery

- The virtual machine was restored to its pre-incident clean state.
- All collected forensic evidence (memory dumps and Volatility reports) was preserved for documentation and future training.
- No residual malicious activity was detected after restoration.

5. Post-Incident Activity (NIST Phase 4)

5.1 Lessons Learned

- Fast-exit PowerShell attacks can bypass traditional monitoring tools due to the short-lived nature of the process.
- Memory forensics is essential for identifying such attacks and reconstructing their behavior.
- The incident response actions were timely and resulted in effective evidence preservation.

5.2 Recommendations

Endpoint Detection and Response (EDR)

Deploy EDR tools capable of detecting:

- PowerShell script block execution
- AMSI bypass attempts
- Suspicious cmd.exe behavior
- Fast-exit execution patterns

Application Control Policies

Enforce AppLocker or Software Restriction Policies to block execution of:

- Unsigned PowerShell scripts
- Scripts executed from user directories or untrusted locations

Logging Improvements

Enable:

- PowerShell Script Block Logging
- Module Logging
- AMSI Deep Content Scanning
- Event ID 4688 (Process Creation Logging)

Evidence Retention

Maintain long-term storage of:

- All BEFORE and AFTER memory dumps
- Volatility report outputs
- Ransom note and encrypted file samples

These materials are valuable for future investigations and academic demonstrations.