DIGITAL FORENSICS & INCIDENT RESPONSE REPORT

Incident Response Investigation Case Study

Case Identifier: WorldCollapsing

Report Date: January 2025

Incident Classification: Supply Chain Attack / Developer Tool Exploitation

Severity Level: CRITICAL

Investigation Status: COMPLETE

INVESTIGATION REPORT - COMPLETED CASE

1. EXECUTIVE SUMMARY

This report documents a comprehensive digital forensic and incident response investigation into a developer workstation compromise discovered during analysis of the WorldCollapsing repository. The incident represents a sophisticated supply-chain style attack that exploited trusted developer tooling mechanisms rather than traditional operating system vulnerabilities.

The attacker leveraged Cursor IDE's Model Context Protocol (MCP) configuration to execute attacker-controlled PowerShell commands automatically when the repository was opened. The malicious activity was introduced through a single unauthorized Git commit authored by an unknown contributor identified as "Luka". The payload was disguised as an image file and executed without explicit user interaction.

KEY INVESTIGATION FINDINGS:

- Exploited vulnerability identified as MCPoison (CVE-2025-54135 and CVE-2025-54136)
- Malicious Git commit hash isolated: c0df0ebeb988e991418029e3021fb7f8542068b2
- Payload successfully analyzed and decoded: images\31.jpg.ps1
- Attack vector fully documented and attributed to known CVE entries
- Comprehensive evidence chain established for forensic purposes

2. INVESTIGATION OBJECTIVES

The following objectives were established to guide this forensic investigation:

1. Determine the mechanism by which initial code execution was achieved on the compromised system
2. Identify all malicious source-code modifications that enabled automatic execution
3. Perform detailed analysis of the payload structure, encoding, and execution behavior
4. Attribute the attack methodology to known vulnerabilities and assign appropriate CVE identifiers
5. Document all findings in accordance with forensic investigation best practices to ensure defensibility

3. ENVIRONMENT OVERVIEW

Attribute	Details
Operating System	Microsoft Windows (Virtual Machine Environment)
Primary User Account	Mizi
Development Tool	Cursor IDE (Integrated Development Environment)
Repository Name	WorldCollapsing
Repository Path	`C:\Users\Mizi\Music\WorldCollapsing`
Incident Classification	Supply-Chain Attack / Developer Tool Exploitation
Attack Surface	Model Context Protocol (MCP) Configuration Files

The affected environment represents a standard developer workstation configuration where source code repositories are implicitly trusted by integrated development environments. This trust relationship forms the foundation of the exploitation methodology employed by the attacker.

4. INITIAL DETECTION & TRIAGE

4.1 Initial Detection Event

The incident was first detected when opening the WorldCollapsing project within Cursor IDE triggered an unexpected User Account Control (UAC) elevation prompt. The prompt requested elevated administrative privileges for an operation attributed to an unfamiliar user account named "Luka".

ANOMALY ASSESSMENT:

UAC prompts during standard IDE initialization procedures are highly irregular and typically indicate configuration-level code execution. This behavior strongly suggested that executable commands were being triggered automatically as part of the project loading sequence, independent of any explicit user action.

4.2 Triage Indicators

The following indicators of compromise were identified during initial triage assessment:

> **INDICATOR 1:** Unexpected UAC elevation prompt during IDE initialization

> **INDICATOR 2:** Request attributed to unknown user account "Luka" not associated with legitimate system operations

> **INDICATOR 3:** Automatic trigger activation during repository loading phase without user-initiated execution

Based on these critical indicators, a comprehensive forensic investigation was immediately initiated with primary focus areas including source code integrity analysis and development tool configuration file examination.

5. REPOSITORY & GIT HISTORY ANALYSIS

5.1 Repository Structure Enumeration

Initial filesystem analysis revealed a Git version control repository named WorldCollapsing located in the non-standard directory path `C:\Users\Mizi\Music\WorldCollapsing`. Superficial examination of the repository contents indicated a collection primarily consisting of image files, suggesting a graphics or media-focused project.

However, the presence of hidden directories including `.git` (version control metadata) and `.cursor` (IDE-specific configuration) warranted comprehensive forensic analysis of repository structure and version history.

5.2 Version Control History Examination

Detailed inspection of the Git commit history revealed a singular anomalous commit authored by an unauthorized contributor identified as "Luka". All remaining commits in the repository history were legitimately attributed to the authorized user account "Mizi".

MALICIOUS COMMIT EVIDENCE:

```
Commit Hash: c0df0ebeb988e991418029e3021fb7f8542068b2
Author: Luka (Unauthorized / Unknown Entity)
Modifications: Configuration files (.cursor\mcp.json)
Added Files: Suspicious executable content (images\31.jpg.ps1)
```

This commit introduced modifications to non-image configuration files and added executable content disguised using media file naming conventions. The combination of an unauthorized commit author with suspicious configuration modifications constituted strong evidence of malicious source-level compromise.

6. MCP CONFIGURATION ABUSE ANALYSIS

6.1 Model Context Protocol Overview

Cursor IDE implements the Model Context Protocol (MCP) as a mechanism for defining tool integrations, server configurations, and automated command execution behavior. MCP configurations are stored as JSON files within the repository structure and are processed during project initialization. By design, these configuration files are implicitly trusted by the IDE without requiring explicit user authorization.

6.2 Malicious Configuration Modifications

Forensic analysis identified that the attacker strategically modified the MCP configuration file located at:

```
File Path: .cursor\mcp.json
File Type: JSON Configuration
Purpose: MCP Server Definition and Command Execution
```

The malicious MCP configuration defined a custom server specification that executed PowerShell commands automatically during IDE initialization. Critical analysis revealed that the configuration explicitly enabled PowerShell execution policy bypass, effectively circumventing Windows security controls designed to prevent unauthorized script execution.

PAYLOAD FILE REFERENCE:

The compromised MCP configuration referenced the following payload file:

```
File Path: images\31.jpg.ps1
Apparent Type: JPEG Image File
Actual Type: PowerShell Executable Script
```

This file was deliberately crafted to appear as a legitimate JPEG image through filename manipulation, while actually containing executable PowerShell code. The dual extension technique represents a social engineering attempt to evade detection during casual repository inspection.

7. PAYLOAD ANALYSIS

7.1 Payload Structure and Obfuscation

The file `31.jpg.ps1` was identified as a malicious PowerShell script employing multiple layers of Base64 encoding for obfuscation purposes. This encoding methodology is commonly utilized by threat actors to evade signature-based detection mechanisms and complicate manual code analysis procedures.

```
Payload Characteristics:
- Multi-layer Base64 encoding scheme
- Runtime self-decoding mechanism
- In-memory execution via Invoke-Expression cmdlet
- Minimal persistent disk artifacts
- Execution policy bypass enabled
```

The script employed runtime self-decoding techniques and executed the decoded payload using the PowerShell `Invoke-Expression` cmdlet. This approach enabled the malicious code to execute entirely within system memory without creating readily identifiable disk-based forensic artifacts.

7.2 Execution Chain Analysis

Upon execution through the MCP configuration during IDE initialization, the following execution sequence was observed:

1. **Stage 1 - Configuration Loading:** Cursor IDE initiates project loading and parses the MCP configuration file (`mcp.json`)
2. **Stage 2 - MCP Server Initialization:** MCP server defined in configuration executes the specified PowerShell script (`31.jpg.ps1`) with execution policy bypass enabled
3. **Stage 3 - Payload Decoding:** PowerShell script performs multi-stage Base64 decoding of embedded payload data

4. **Stage 4 - Code Execution:** Decoded content executes within the current user security context
5. **Stage 5 - Secondary Compromise:** Additional stages of system compromise are initiated (documented separately in Incident Response Report 2)

The payload was intentionally designed to evade casual inspection by mimicking the naming conventions and directory structure of legitimate repository contents. The placement within an "images" directory and the use of a JPEG-suggesting filename enabled the malicious file to blend seamlessly into the repository's apparent purpose as an image collection.

8. VULNERABILITY ATTRIBUTION

8.1 Vulnerability Classification

The attack methodology documented in this investigation aligns with MCP configuration poisoning, a vulnerability class commonly designated as "MCPoison". This vulnerability enables attacker-controlled MCP configuration files embedded within otherwise trusted repositories to execute arbitrary commands automatically, without requiring explicit user consent or interaction beyond the standard action of opening a project.

8.2 CVE Assignment

The following Common Vulnerabilities and Exposures (CVE) identifiers have been assigned to document this vulnerability:

ASSIGNED CVE IDENTIFIERS:

- **CVE-2025-54135:**

  Improper trust and insufficient validation of MCP configuration files in Cursor IDE
- **CVE-2025-54136:**

  Automatic code execution without user approval or sandboxing in MCP server implementation

These CVE entries formally document the security deficiencies in Cursor IDE's handling of MCP configuration files, specifically the lack of proper validation, sandboxing, and user authorization mechanisms prior to executing commands specified in repository-embedded configuration files.

9. IMPACT ASSESSMENT

INCIDENT IMPACT SUMMARY:

EXECUTION TIMING:	Unauthorized command execution occurred automatically during repository initialization
EXPLOITATION REQUIREMENTS:	No operating system vulnerabilities required; attack leveraged application-level trust relationships
USER INTERACTION LEVEL:	Minimal interaction required; compromise triggered by standard action of opening project
ATTACK CLASSIFICATION:	High-severity supply-chain attack targeting software development personnel
PRIVILEGE CONTEXT:	Initial execution in user context with attempted elevation to administrative privileges
DETECTION DIFFICULTY:	High; payload disguised as legitimate repository content with minimal security indicators

This attack vector represents a significant threat to software development supply chain integrity. The exploitation methodology demonstrates that developers who clone, download, or otherwise obtain compromised repositories are immediately vulnerable to system compromise upon performing the routine action of opening the project within their development environment. The attack requires no traditional security indicators, social engineering prompts, or explicit user authorization beyond standard development workflow actions.

10. CONCLUSIONS AND RECOMMENDATIONS

This incident investigation conclusively demonstrates the security risks inherent in implicit trust relationships between integrated development environments and repository configuration files. The attacker successfully achieved reliable, automatic code execution without exploiting traditional software vulnerabilities, operating system security controls, or employing explicit social engineering techniques.

The attack succeeded due to the convergence of multiple security deficiencies:

- Implicit trust of repository-embedded configuration files by Cursor IDE without integrity verification
- Absence of sandboxing or permission prompting mechanisms for MCP server execution
- Configuration-level ability to bypass PowerShell execution policies and security controls
- Effective use of social engineering through filename masquerading and directory placement
- Lack of commit author verification or repository integrity checking mechanisms

RECOMMENDATIONS FOR PREVENTION AND MITIGATION:

1. **CONFIGURATION FILE AUDITING:**

   Implement mandatory inspection procedures for `.cursor`, `.vscode`, and similar IDE configuration directories prior to project initialization

2. **AUTOMATIC EXECUTION CONTROLS:**

   Disable automatic command execution in development tools or implement user authorization prompts for all configuration-defined operations

3. **DEVELOPMENT ENVIRONMENT HARDENING:**

   Apply security controls and monitoring typically reserved for production systems to development workstations

4. **COMMIT AUTHOR VERIFICATION:**

   Establish procedures to investigate and verify unfamiliar or unexpected commit authors before incorporating changes

5. **CODE SIGNING REQUIREMENTS:**

   Implement mandatory code signing for repository commits from trusted, verified developers

6. **SANDBOXING IMPLEMENTATION:**

   Require IDE-executed commands to operate within restricted sandboxed environments

7. **SECURITY AWARENESS TRAINING:**

   Educate development personnel on supply-chain attack vectors targeting developer tooling

11. RECOVERED EVIDENCE

The investigation successfully recovered the following forensic evidence flag, which encodes the key components of the attack methodology:

```
nite{CVE-2025-54135/6_c0df0ebeb988e991418029e3021fb7f8542068b2_31.jpg.ps1}
```

EVIDENCE FLAG COMPONENT ANALYSIS:

- **CVE-2025-54135/6:**

  Identified vulnerability CVE identifiers documenting the MCP configuration poisoning vulnerability

- **C0DF0EBEB988E991418029E3021FB7F8542068B2:**

  Git commit hash of the malicious commit introducing the exploit configuration

- **31.JPG.PS1:**

  Filename of the payload file containing the obfuscated PowerShell script

---

END OF REPORT

Document Classification: Investigation Complete

Report Generated: January 2025

Challenge Designation: Incident Response 1 - WorldCollapsing Case

Investigation Status: CLOSED