SOC Playbook: Process Injection Detection (T1055)

1. Objective

Detect and respond to malicious **process injection** techniques such as DLL injection, reflective DLL loading, APC injection, process hollowing, or shellcode injection, often used by adversaries to execute payloads stealthily or evade security controls.

2. Scope

- Monitor injection behaviors across **Windows, Linux, macOS** endpoints.
- Detect usage of low-level system APIs for memory allocation, code writing, and remote thread creation.
- Identify suspicious parent-child process combinations, memory manipulation, and EDR bypass behavior.
- Prevent further execution and initiate containment and investigation workflows.

3. Log Sources

Platform	Log Source	Description
Windows	Windows	Windows
Sysmon (Event ID 8, 10, 1)	Sysmon (Event ID 8, 10, 1)	Sysmon (Event ID 8, 10, 1)
Image loading, remote thread creation, process start	Image loading, remote thread creation, process start	Image loading, remote thread creation, process start
Windows	Windows	Windows
Security Logs (4688)	Security Logs (4688)	Security Logs (4688)
Process creation	Process creation	Process creation

4. Detection Rules / Alerts

Alert Name	Description	Triggers / Examples
Remote Thread Injection	One process creates a thread in another process	Sysmon Event ID 8 (CreateRemoteThread)
Suspicious Memory Allocation	High-entropy memory regions with execute rights	Use of VirtualAllocEx, WriteProcessMemory, NtProtectVirtualMemory
Process Hollowing Detected	Parent spawns process and overwrites memory	Process starts suspended, then memory is replaced and resumed

Unusual API Sequence	Use of known injection API sequence	OpenProcess → VirtualAllocEx → WriteProcessMemory → CreateRemoteThread
Injection into System Process	Injection into lsass.exe, explorer.exe, etc.	Rare for normal applications to inject into system processes
EDR Bypass Patterns	Known patterns of AMSI bypass or API unhooking	Strings like AmsiScanBuffer, ETWTI, memcpy trampoline in logs
Suspicious Image Load	Unexpected DLL loaded by unusual process	Sysmon Event ID 7, DLL injected via AppInit_DLLs, etc.

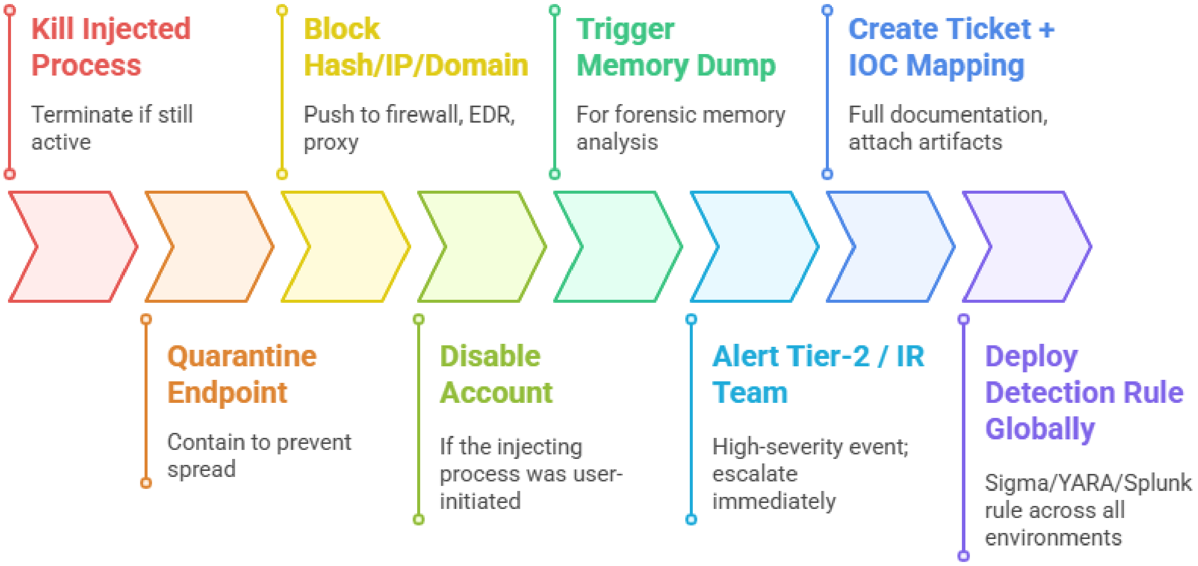
5. Automated Enrichment

Enrichment Task	Description
User and Host Info	User and Host Info
Who initiated the injection and on which host	Who initiated the injection and on which host
Injected Process	Injected Process
Target process, process ID, hash, command line	Target process, process ID, hash, command line
Injection Tool Detection	Injection Tool Detection
Check if injecting process is LOLBin or known malware	Check if injecting process is LOLBin or known malware

6. Automated Response Play

Step	Action
1. Kill Injected Process	Terminate if still active
2. Quarantine Endpoint	Contain to prevent spread
3. Block Hash/IP/Domain	Push to firewall, EDR, proxy
4. Disable Account	If the injecting process was user-initiated
5. Trigger Memory Dump	For forensic memory analysis
6. Alert Tier-2 / IR Team	High-severity event; escalate immediately
7. Create Ticket + IOC Mapping	Full documentation, attach artifacts
8. Deploy Detection Rule Globally	Sigma/YARA/Splunk rule across all environments

Comprehensive Incident Response Timeline



Kill Injected Process
Terminate if still active

Quarantine Endpoint
Contain to prevent spread

Block Hash/IP/Domain
Push to firewall, EDR, proxy

Disable Account
If the injecting process was user-initiated

Trigger Memory Dump
For forensic memory analysis

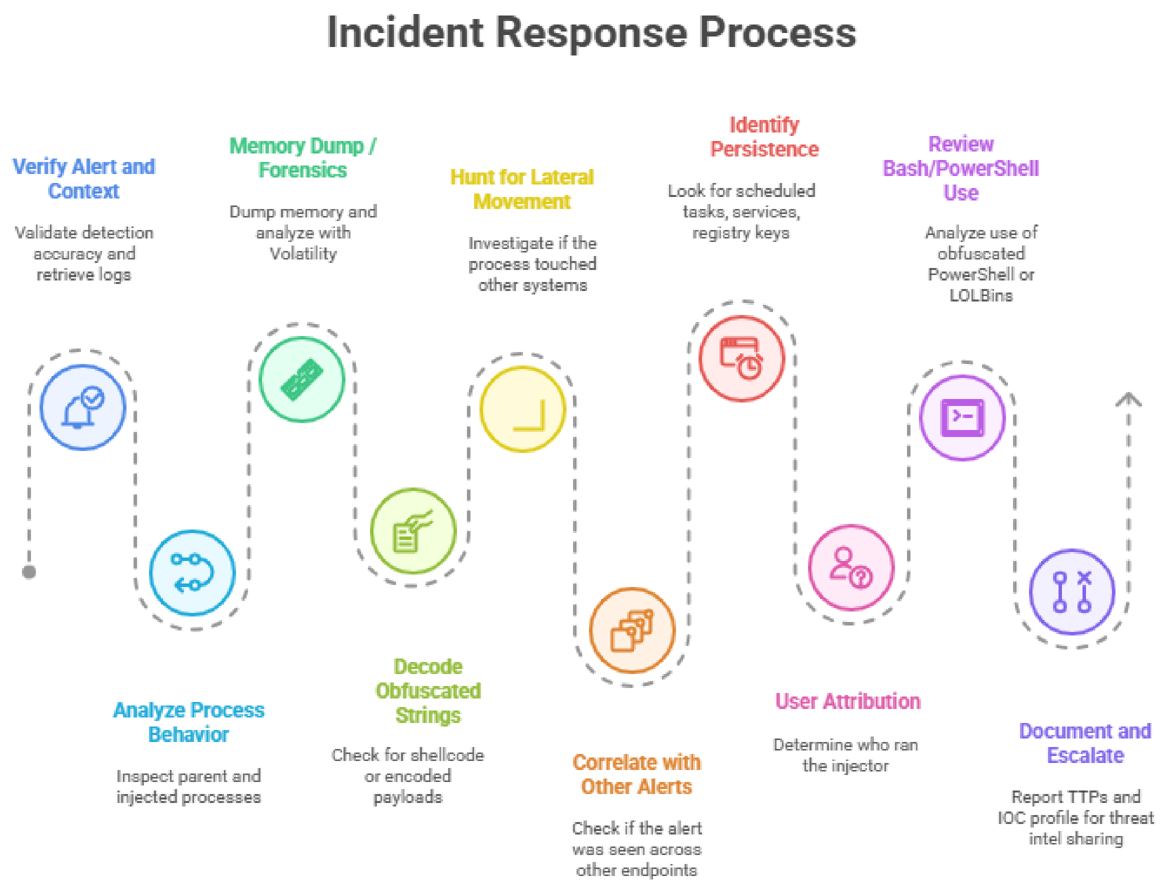
Alert Tier-2 / IR Team
High-severity event; escalate immediately

Create Ticket + IOC Mapping
Full documentation, attach artifacts

Deploy Detection Rule Globally
Sigma/YARA/Splunk rule across all environments

7. Investigation Checklist

Step	Description
1. Verify Alert and Context	Validate detection accuracy and retrieve all logs
2. Analyze Process Behavior	Inspect parent and injected processes, privileges
3. Memory Dump / Forensics	If possible, dump memory of injected process and analyze with Volatility
4. Decode Obfuscated Strings	Check if shellcode or encoded payloads are embedded
5. Hunt for Lateral Movement	Did injected process touch other systems or escalate?
6. Correlate with Other Alerts	Was this seen across other endpoints?
7. Identify Persistence	Look for scheduled tasks, services, registry keys
8. User Attribution	Who ran the injector or allowed it to run
9. Review Bash/PowerShell Use	Many injectors use obfuscated PowerShell or LOLBins
10. Document and Escalate	Report full TTPs and IOC profile for threat intel sharing



Incident Response Process
Verify Alert and Context
Validate detection accuracy and retrieve logs
Analyze Process Behavior
Inspect parent and injected processes
Memory Dump / Forensics
Dump memory and analyze with Volatility
Decode Obfuscated Strings
Check for shellcode or encoded payloads
Hunt for Lateral Movement
Investigate if the process touched other systems
Correlate with Other Alerts
Check if the alert was seen across other endpoints
Identify Persistence
Look for scheduled tasks, services, registry keys
User Attribution
Determine who ran the injector
Review Bash/PowerShell Use
Analyze use of obfuscated PowerShell or LOLBins
Document and Escalate
Report TTPs and IOC profile for threat intel sharing

8. Playbook Notes

- Monitor API abuse patterns with behavioral analytics.
- Block known LOLBins (e.g., rundll32.exe, mshta.exe) from suspicious paths.
- Educate analysts on shellcode and injection analysis using tools like Cuckoo, PEStudio.
- Enable full command-line logging and image load monitoring via Sysmon.
- Use memory forensics when possible — injectors often leave little disk evidence.