

AI-Driven Incident Response

Combining established IR frameworks with AI-powered analysis to detect, contain, and recover from security incidents faster.

Why AI Enhances Incident Response

Security operations centers face a relentless volume of alerts. The average SOC receives over 10,000 alerts per day, yet analyst teams can realistically investigate only a fraction of them. This gap between signal volume and human capacity is where incidents slip through. Attackers know this; many offensive playbooks rely on burying malicious activity inside a flood of benign noise.

AI fundamentally changes this equation. Large language models can triage thousands of alerts in seconds, correlate indicators of compromise (IOCs) across disparate log sources, draft containment actions, and generate post-incident reports that would take a human analyst hours to write. The key advantages are:

- **Speed:** AI reduces mean-time-to-detect (MTTD) and mean-time-to-respond (MTTR) by automating the most time-consuming phases of the IR lifecycle.
- **Consistency:** AI applies the same analytical rigor to alert number 10,000 as it does to alert number 1. Human fatigue does not degrade its performance.
- **Scalability:** A single AI-augmented analyst can handle the workload that previously required an entire team.
- **Knowledge synthesis:** LLMs can instantly cross-reference IOCs against known threat intelligence, map TTPs to MITRE ATT&CK, and surface relevant context from prior incidents.

AI does not replace the incident responder. It amplifies them. The human provides judgment, authorization, and institutional context; the AI handles speed, breadth, and tireless execution of repetitive analytical tasks.

The NIST SP 800-61 IR Framework

The National Institute of Standards and Technology published SP 800-61 (Computer Security Incident Handling Guide) as the foundational reference for incident response. It defines four phases:

1. **Preparation:** Establishing policies, plans, tools, and training before incidents occur. This includes building playbooks, setting up communication channels, and ensuring logging infrastructure is in place.
2. **Detection and Analysis:** Identifying that an incident has occurred, determining its scope, and classifying its severity. This is where alert triage, log analysis, and IOC correlation happen.
3. **Containment, Eradication, and Recovery:** Stopping the incident from spreading, removing the attacker's foothold, and restoring systems to normal operation.
4. **Post-Incident Activity:** Documenting lessons learned, updating playbooks, and tracking metrics to improve future response.

NIST emphasizes that these phases are not strictly linear. Detection may trigger immediate containment actions, and post-incident findings often feed back into improved preparation. The framework is a cycle, not a waterfall.

The SANS 6-Step IR Process

The SANS Institute expands the NIST model into six discrete steps, which provide more granular guidance for practitioners:

1. **Preparation** — Build the IR team, establish policies, deploy tools
2. **Identification** — Detect and validate the incident
3. **Containment** — Limit the blast radius
4. **Eradication** — Remove the threat actor and persistence mechanisms
5. **Recovery** — Restore systems and verify integrity
6. **Lessons Learned** — Document findings and improve processes

The primary difference from NIST is that SANS treats containment, eradication, and recovery as three separate phases rather than one combined phase. This distinction is practically useful because each step has different tools, decision criteria, and risk profiles. We will walk through all six phases below.

Phase 1: Preparation

Preparation is the foundation that determines whether your team fumbles or executes when an incident strikes. Key deliverables include:

- **Incident Response Plan (IRP):** A formal document defining roles, escalation paths, severity classifications, and communication procedures.
- **Playbooks:** Step-by-step runbooks for common incident types (ransomware, data exfiltration, phishing compromise, DDoS).
- **Communication Templates:** Pre-drafted notifications for executives, legal, customers, and regulators. During an active incident, writing from scratch wastes precious time.
- **Tooling:** EDR agents deployed, SIEM ingesting logs, forensic workstations imaged, network capture capability tested.

```
# IR checklist template generator
ir_checklist = {
    "preparation": [
        "IR plan reviewed and approved within last 90 days",
        "Contact list (legal, exec, PR, law enforcement) verified",
        "Playbooks exist for: ransomware, phishing, data breach, DDoS",
        "Forensic toolkit tested: disk imaging, memory capture, log export",
        "Communication templates drafted for each severity level",
        "Tabletop exercise conducted within last 6 months",
    ],
    "detection_tools": [
        "SIEM receiving logs from all critical systems",
        "EDR deployed on all endpoints",
        "Network IDS/IPS active and tuned",
        "Threat intel feeds integrated into alert pipeline",
    ],
}

def print_checklist(checklist: dict) -> None:
    for phase, items in checklist.items():
        print(f"\n=== {phase.upper().replace('_', ' ')} ===")
        for item in items:
            print(f"  [ ] {item}")
```

Phase 2: Detection and Identification

Detection is where most IR processes begin in practice. Alerts arrive from SIEM rules, EDR detections, user reports, or external notifications (threat intel partners, law enforcement). The challenge is separating real incidents from false positives and determining severity quickly.

AI-assisted detection focuses on three areas:

- **Alert Triage:** An LLM can consume a batch of raw alerts and classify them by likelihood of being true positives, grouping related alerts into clusters.
- **Indicator Correlation:** Given a suspicious IP, hash, or domain, AI can cross-reference it against threat intelligence databases, historical logs, and known attack patterns.
- **Contextual Analysis:** AI can explain what a detection rule triggered on, what the normal baseline looks like, and why this event is anomalous.

```
import re
from datetime import datetime

def extract_iocs(text: str) -> dict:
    """Extract indicators of compromise from unstructured text."""
    iocs = {
        "ipv4": re.findall(
            r'\b(?:\d{1,3}\.){3}\d{1,3}\b', text
        ),
        "domains": re.findall(
            r'\b(?:[a-zA-Z0-9-]+\.[a-zA-Z]{2,})\b', text
        ),
        "md5": re.findall(r'\b[a-fA-F0-9]{32}\b', text),
        "sha256": re.findall(r'\b[a-fA-F0-9]{64}\b', text),
        "emails": re.findall(
            r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text
        ),
        "cves": re.findall(r'CVE-\d{4}-\d{4,7}', text),
    }
    # Deduplicate each list
    return {k: list(set(v)) for k, v in iocs.items() if v}

# Example usage with a threat report excerpt
report = """
```

The attacker connected from 198.51.100.23 and 203.0.113.42.
Malware hash: 5d41402abc4b2a76b9719d911017c592

Phase 3: Containment

Once an incident is confirmed, the immediate priority is stopping the bleeding. Containment strategies fall into two categories:

- **Short-term containment:** Immediate actions to stop active damage. Isolate the affected host from the network, block the attacker's IP at the firewall, disable compromised accounts. These actions are taken within minutes.
- **Long-term containment:** Sustainable measures that allow investigation to continue while preventing re-compromise. This may include moving affected systems to an isolated VLAN, deploying additional monitoring, or applying temporary firewall rules that permit forensic access while blocking external communication.

```
# Short-term containment commands

# Isolate host via firewall (block all except management)
iptables -I INPUT -s 0.0.0.0/0 -j DROP
iptables -I INPUT -s 10.0.0.0/8 -p tcp --dport 22 -j ACCEPT
iptables -I OUTPUT -d 0.0.0.0/0 -j DROP

# Block malicious IP at network perimeter
firewall-cmd --permanent --add-rich-rule='rule family="ipv4" \
    source address="198.51.100.23" drop'
firewall-cmd --reload

# Disable compromised user account
usermod -L compromised_user
passwd -l compromised_user

# Kill suspicious process by PID
kill -9 $(pgrep -f "suspicious_binary")
```

Preserve evidence: Before wiping or reimaging a compromised system, capture a forensic disk image and memory dump. Containment must not destroy the evidence needed for eradication and post-incident analysis.

Phase 4: Eradication

Eradication goes beyond containment: it removes the root cause and all persistence mechanisms the attacker has installed. Common eradication tasks include:

- Removing malware binaries, scripts, and backdoor accounts
- Cleaning persistence mechanisms: cron jobs, systemd services, registry run keys, startup scripts
- Patching the vulnerability that enabled initial access
- Rotating all credentials that may have been exposed (passwords, API keys, certificates)
- Verifying that no additional systems were compromised laterally

```
def check_persistence_mechanisms(host: str) -> dict:
    """Check common Linux persistence locations for anomalies."""
    import subprocess

    checks = {
        "cron_jobs": "crontab -l 2>/dev/null; ls -la /etc/cron.* / 2>/dev/null",
        "systemd_services": (
            "systemctl list-unit-files --type=service --state=enabled "
            "| grep -v vendor"
        ),
        "authorized_keys": "find /home -name authorized_keys -exec cat {} \\\;",
        "bashrc_modifications": (
            "find /home -name '.bashrc' -newer /etc/os-release"
        ),
        "unusual_suid": "find / -perm -4000 -type f 2>/dev/null",
    }

    results = {}
    for name, cmd in checks.items():
        result = subprocess.run(
            cmd, shell=True, capture_output=True, text=True
        )
        results[name] = result.stdout.strip()
    return results
```

Phase 5: Recovery

Recovery is the process of restoring affected systems to normal operation and verifying that the threat has been fully eliminated. This phase requires careful validation to avoid restoring a system that still contains attacker artifacts.

- **System restoration:** Rebuild from known-good images or backups taken before the compromise window. Never restore from backups made after the attacker gained access without thorough inspection.
- **Monitoring:** Deploy enhanced monitoring on recovered systems for at least 30 days. Watch for signs of re-compromise: unexpected outbound connections, new user accounts, modified system files.
- **Validation:** Run integrity checks (file hashes, configuration audits) against known-good baselines. Verify that all patching and credential rotation completed successfully.

Staged recovery: Bring systems back online in stages, starting with the least critical. Monitor each stage before proceeding. A phased approach limits blast radius if eradication was incomplete.

Phase 6: Post-Incident Activity

The post-incident phase is where organizations build institutional memory. Without disciplined documentation and retrospective analysis, the same mistakes recur and response times never improve.

Lessons Learned

Conduct a blameless post-mortem within 5 business days of incident closure. Key questions: What happened? When did we detect it? What slowed us down? What worked well? What should we change?

Report Writing

AI is exceptionally useful for drafting incident reports. Given a timeline of events, IOCs, and actions taken, an LLM can produce a structured report in minutes.

```
def generate_timeline(events: list[dict]) -> str:
    """Generate a formatted incident timeline from event records."""
```

```
lines = ["# Incident Timeline", ""]
for event in sorted(events, key=lambda e: e["timestamp"]):
    ts = event["timestamp"]
    src = event.get("source", "unknown")
    desc = event["description"]
    lines.append(f"| {ts} | {src} | {desc} |")

header = "| Timestamp | Source | Event |\n|---|---|---|"
lines.insert(2, header)
return "\n".join(lines)

# Example events
events = [
    {"timestamp": "2026-04-09T02:14:00Z", "source": "SIEM",
     "description": "Alert: unusual outbound traffic to 198.51.100.23"},
    {"timestamp": "2026-04-09T02:18:00Z", "source": "EDR",
     "description": "Malicious process detected: /tmp/.hidden/beacon"},
    {"timestamp": "2026-04-09T02:22:00Z", "source": "Analyst",
     "description": "Host isolated from network, containment initiated"},
    {"timestamp": "2026-04-09T03:45:00Z", "source": "Forensics",
     "description": "Root cause identified: CVE-2024-21887 exploit via VPN"},
    {"timestamp": "2026-04-09T06:00:00Z", "source": "IR Lead",
     "description": "Eradication complete, recovery phase initiated"},
]
```

Metric Tracking

Track key metrics across incidents to measure and improve your IR capability over time:

Metric	Definition	Target
MTTD	Mean time from compromise to detection	< 24 hours
MTTR	Mean time from detection to full recovery	< 72 hours
Containment time	Time from detection to initial containment	< 1 hour
False positive rate	Percentage of investigated alerts that are benign	< 30%
Recurrence rate	Incidents caused by the same root cause within 90 days	0%

AI-Assisted IR with Claude

Claude can serve as an IR co-pilot across every phase of the process. Practical applications include:

- **IOC analysis:** Paste a block of raw log data and ask Claude to extract all IOCs, classify them, and suggest threat intel lookups.
- **Drafting communications:** Provide incident details and ask Claude to draft executive notifications, customer disclosures, or regulatory filings appropriate to the severity level.
- **Timeline generation:** Feed Claude timestamped event data and receive a formatted incident timeline with causal annotations.
- **Playbook execution:** Describe the incident type and ask Claude to walk through the relevant playbook step by step, adapting recommendations to your specific environment.
- **Post-mortem facilitation:** Provide raw notes from the incident and ask Claude to structure them into a lessons-learned document with actionable recommendations.

Operational security: Never paste sensitive production data (credentials, PII, unredacted customer records) into any external AI service during an active incident. Use redacted or sanitized data, or deploy a self-hosted model for sensitive analysis.

Further Reading

- [NIST SP 800-61 Rev. 2 — Computer Security Incident Handling Guide](#)
- [SANS Incident Handler's Handbook](#)
- [MITRE ATT&CK Framework](#)
- [Anthropic Docs — Tool Use \(for building IR automation agents\)](#)

Download Files

These files contain synthetic incident data for the IR challenges. All files are plain text (.txt) and safe to open.

↓ [incident-alerts.txt](#) ~9 KB

↓ [affected-systems.txt](#) ~10 KB

↓ [ioc-list.txt](#) ~9 KB

↓ [ir-comms-template.txt](#) ~16 KB