CHAPTER 14

Proactive Activities

Chapter Overview

01

Threat Hunting

Assume breach — hypothesis-driven search for undetected adversaries

02

Adversary Emulation

Atomic Red Team, CALDERA, and purple team exercises

Threat Hunting

Assume breach — go find them

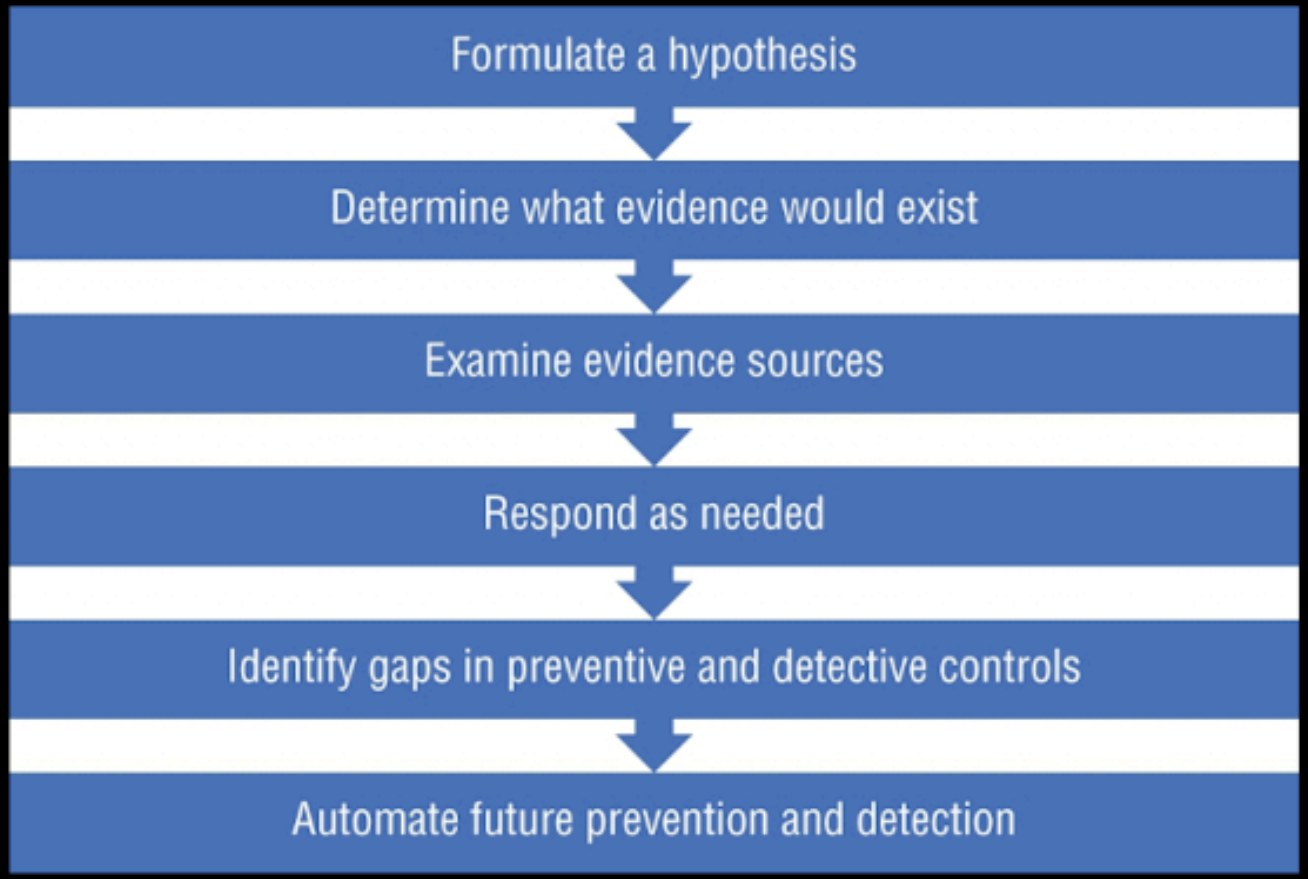

Figure 14.1: The threat-hunting process

Threat Hunting Fundamentals

Threat hunting: proactively search for adversaries that evaded automated detection

Core assumption: you are already compromised — the question is where and for how long

Alert-driven response is reactive; hunting is proactive — different mindset entirely

MITRE ATT&CK: every technique is a potential hunt hypothesis (1400+ techniques)

Hunt loop: Form hypothesis → Identify data sources → Execute hunt → Analyze → Document → Automate

Output: either 'found something' (escalate to IR) or 'tuned detection' (new detection rule)

Maturity: hunts that find nothing still produce value — they become detection content

Building Hunt Hypotheses

Hypothesis format: 'Attackers using [technique] would leave [indicator] in [data source]'

Example: 'Attackers using Kerberoasting would generate 4769 events with RC4 encryption'

Example: 'C2 beaconing via HTTP would appear as regular-interval connections in proxy logs'

Threat intelligence: recent TTPs from threat reports → immediate hunt hypotheses

Crown jewel analysis: what data is most valuable? Work backwards from target to access path

Gap analysis: which ATT&CK techniques have NO detection coverage? Hunt those first

Prior incidents: the attacker may still be present — use the IR timeline as a hunt guide

Process & Memory Hunt Examples

Process-Based Hunts

Parent-child anomalies: winword.exe → powershell.exe (T1566, T1059)

LOLBins with networking: certutil.exe -urlcache, regsvr32.exe, mshta.exe

Long-lived PowerShell: processes running >8 hours without user interaction

Hollow processes: process image path doesn't match loaded module on disk

Unusual LSASS access: non-system processes with PROCESS_VM_READ on lsass

Memory & Injection Hunts

Executable memory outside image-backed regions (malfind in Volatility)

DLLs loaded from unusual paths: AppData, Temp, ProgramData

Threads with start address pointing to heap or anonymous memory

Process with network connection but no corresponding listening socket

svchost.exe with no -k argument in command line

Network & Credential Hunt Examples

Network-Based Hunts

Beaconing: automated C2 check-ins at fixed intervals (1–60 min jitter)

Long connections: hours-long sessions on port 443 with low byte transfer

DNS tunneling: high-entropy subdomains, unusually large TXT record responses

Newly registered domains: C2 domains registered within last 30 days

Internal east-west: workstation → workstation on SMB — never normal

Credential Abuse Hunts

4769 RC4 spike: bulk TGS requests from single host in short window

4648 Explicit credential use: lateral movement tools trigger this constantly

Admin account logons on workstations other than their assigned host

Service accounts logging on interactively (Type 2) — should never happen

NTLM Type 3 responses from hosts that should be using Kerberos only

Data Sources & Hunt Infrastructure

EDR telemetry: process, file, registry, network events at endpoint level

SIEM / log aggregation: centralized query across all hosts simultaneously

DNS logs: full query/response history — gold mine for C2 and data exfil hunting

Proxy / web gateway logs: outbound HTTP/HTTPS URL and byte-count details

NetFlow / IPFIX: connection metadata without full payload — covers the whole network

Zeek logs: rich protocol-specific fields from network taps

Velociraptor VQL: run any hunt query across all endpoints simultaneously in real time

Key principle: if the data source doesn't exist, the hunt can't run — build visibility first

Measuring Hunt Effectiveness

MTTD (Mean Time to Detect): how long attackers dwell before discovery — aim to reduce it

Detection coverage: percentage of ATT&CK techniques with at least one active detection

Hunt yield: ratio of investigations opened per hunt executed — track over time

New detections: every hunt that finds something should produce at least one new detection rule

False positive rate: high FPR burns analyst time — tune aggressively

Hunt cadence: high-priority techniques should be hunted weekly; others monthly/quarterly

Document everything: hypothesis, data source, query, results, and disposition

Hunt Platform Setup & Tooling

Minimum viable hunt stack: SIEM + EDR + DNS logs + NetFlow

Elastic Stack: free, scalable, Kibana for visualization — most common open-source stack

Splunk: market-leading SIEM — powerful query language (SPL), high cost at scale

Microsoft Sentinel: cloud-native SIEM — deep Azure/M365 integration, pay per GB

Velociraptor: complements SIEM with live endpoint query capability

Threat intel integration: MISP or commercial TI feed populates indicators for hunt queries

SOAR platforms: orchestrate repetitive hunt steps (XSOAR, Splunk SOAR)

Start simple: 5 good hunt queries executed weekly beats 100 automations never reviewed

MITRE ATT&CK-based threat validation — mapping real-world attacks to the kill chain

Adversary Emulation

Test your defenses before attackers do

Adversary Emulation vs Red Team vs Pentest

Penetration test: find vulnerabilities — scope is systems, goal is access

Red team: simulate full adversary campaign — scope is detection & response capability

Adversary emulation: execute specific TTPs to test specific detections — scoped, collaborative

Purple team: red + blue working together with immediate feedback — learning exercise

Key advantage of emulation: blue team learns what detection gaps exist and fixes them in real time

Prerequisites: logging must be in place before emulation — otherwise nothing to measure

ATT&CK Navigator: visualize coverage gaps before planning an emulation exercise

Atomic Red Team

Framework Overview

Open-source library mapping to every ATT&CK technique (1400+ atomic tests)

Each test: Prerequisites, Executor, Cleanup — self-contained and reversible

PowerShell framework: Install-AtomicRedTeam; Import-Module Invoke-AtomicRedTeam

Invoke-AtomicTest T1059.001: runs all PowerShell execution tests

Invoke-AtomicTest T1059.001 -TestNumbers 1: runs a single specific test

Using Atomics in Purple Team

Step 1: choose ATT&CK technique with weak/no detection coverage

Step 2: run atomic test → observe what telemetry is generated

Step 3: write SIEM/EDR detection rule targeting the observed artifacts

Step 4: run atomic again → verify the detection fires

Step 5: add test to regression suite — re-run after every detection change

MITRE CALDERA

Platform & Architecture

Automated adversary emulation platform from MITRE — open source

Server: Python-based web application with REST API and UI

Agents (Sandcat): deployed to target systems, beacon to CALDERA server

Abilities: individual TTPs mapped to ATT&CK — hundreds included

Adversary profiles: ordered list of abilities = simulated threat actor

Running an Exercise

Step 1: deploy Sandcat agent on test endpoint (staged, not production)

Step 2: select adversary profile (e.g., APT3, ransomware, credential theft)

Step 3: start operation — CALDERA executes each ability, records results

Step 4: review operation report — which abilities succeeded vs. blocked

Step 5: for each undetected success, create detection and re-run

Closing the Loop — From Hunt to Detection

Every successful hunt finding becomes a permanent detection rule in the SIEM or EDR

Every successful emulation finding that wasn't detected becomes an alert to build

Document the ATT&CK technique, data source, query, and false positive rate for each rule

Detection-as-code: store detection logic in version control alongside the code it monitors

Regression testing: run atomic tests after every detection platform change to verify nothing broke

Threat intel integration: ingest new TTP reports → update hunt backlog → emulate within 30 days

The loop: Threat Intel → Hunt Hypothesis → Emulate → Detect → Hunt again

Reporting Hunt Findings

Audience: two reports — executive (1 page) and technical (as detailed as needed)

Executive report: threats found, dwell time, business risk, recommended actions

Technical report: hypothesis, data sources, query logic, findings, IOCs, investigation steps

Negative hunts: 'found nothing' is a valid result — document what was looked at

ATT&CK mapping: include technique IDs for all findings — enables coverage tracking over time

IOC dissemination: share file hashes, IPs, and domains to SIEM/EDR blocking within 24 hours

Trend reporting: monthly summary — hunts run, yield rate, new detections created

Value demonstration: translate technical findings into business risk for leadership buy-in

Integrating Hunting with Incident Response

Detection gap: every IR alert starts as a hunt hypothesis — close the loop

During IR: while one team contains, another hunts for additional compromised systems

Hunt feeds IR: hypotheses proven true during hunting are handed to IR for full investigation

IR feeds hunts: TTPs discovered during IR become hunt hypotheses for the next quarter

Post-incident hunt: after remediation, re-hunt using attacker's known TTPs — verify full eviction

Shared tooling: hunt and IR teams share the same SIEM, EDR, and log infrastructure

Joint exercises: purple team exercises train both hunt and IR capabilities simultaneously

Continuous loop: Incident → Document TTPs → Hunt for TTPs → Detect → Improve

Conclusion

Reactive-only security guarantees you discover breaches long after damage is done

Threat hunting systematically reduces attacker dwell time — every day of hunting saves months of cleanup

MITRE ATT&CK provides a universal language and priority framework for both hunts and emulations

Atomic Red Team turns ATT&CK into executable tests — your detection coverage is only as good as what you've tested

CALDERA automates multi-stage adversary campaigns — stress tests detection and response together

Complete the loop: intelligence → hypothesis → hunt → emulate → detect → repeat

Knowledge Check