1. **TheHive — High-Priority Case (Unauthorized Access → escalate to Tier 2)**

**1.1 TheHive case JSON (POST payload)**

Use this JSON with your TheHive API (POST /api/case). Replace THEHIVE\_URL and THEHIVE\_APIKEY with your values.

{

"title": "Unauthorized Access — Server-Y (192.168.204.137)",

"description": "Detected unauthorized access on Server-Y (Debian 22.x, 192.168.204.137) on 2025-08-18 13:00. Indicators show successful authentication consistent with MITRE T1078. Initial containment: network isolation applied and session termination. Evidence captured: /var/log/auth.log extracts, process lists, and pcap snippets. Case escalated to Tier 2 for full forensic investigation, credential verification, and lateral movement assessment.",

"severity": 3,

"tlp": 2,

"tags": ["T1078", "Unauthorized Access", "Server-Y"],

"artifacts": [

{ "dataType": "ip", "data": "192.168.204.137", "message": "Compromised host (Server-Y)" },

{ "dataType": "ip", "data": "192.168.204.138", "message": "Related workstation/agent" }

],

"status": "New",

"owner": "SOC Tier 1"

}

**1.2 cURL example to create the case**

curl -X POST "https://THEHIVE\_URL/api/case" \

-H "Content-Type: application/json" \

-H "Authorization: Bearer YOUR\_THEHIVE\_APIKEY" \

-d @thehive\_case.json

(put the JSON into thehive\_case.json before running.)

**1.3 Escalation action (UI / API)**

* UI: open the case → set **Owner** to Tier 2 or add a Task assigned to the Tier 2 team; add the Tier 2 escalation note (see 100-word summary below).
* API: update case owner / add a task. Example (pseudo):

# Assign to Tier 2 (replace <caseId> and "Tier 2" with your team name)

curl -X PATCH "https://THEHIVE\_URL/api/case/<caseId>" \

-H "Authorization: Bearer YOUR\_THEHIVE\_APIKEY" \

-H "Content-Type: application/json" \

-d '{"owner":"Tier 2","status":"In Progress"}'

**2) Tier-2 Escalation Summary (exactly 100 words)**

On 2025-08-18 at 13:00 UTC, monitoring detected unauthorized access on Server-Y (192.168.204.137) consistent with MITRE T1078. Initial triage confirmed a successful SSH authentication followed by abnormal privilege escalation attempts and unusual outbound connections. Immediate containment included isolating the host from the network and terminating suspicious sessions; forensic artifacts (auth logs, process lists, pcap snippets) were preserved. Evidence collection prioritized volatile memory, authentication logs, and network captures. This case is escalated to Tier Two for in-depth malware/credential analysis, lateral movement discovery, credential resets where required, and remediation guidance including multi-factor enforcement and transient access controls. Tier Two to report findings within 8 hours.

(100 words — counted.)

**3) SITREP (Google Docs ready — full detailed report)**

You can copy-paste the block below directly into a Google Doc. It’s organized so an executive and technical audience get the right level of detail.

Title: Unauthorized Access on Server-Y

Incident ID: [auto-assigned]

Detected: 2025-08-18 13:00 (UTC)

Affected Host: Server-Y (Debian 22.x) — 192.168.204.137

Related Host / Agent: Windows 11 VM — 192.168.204.138

MITRE ATT&CK: T1078 (Valid Accounts)

Priority: High

Summary:

At 2025-08-18 13:00 monitoring systems flagged a successful SSH authentication on Server-Y (192.168.204.137) that correlated with a burst of privileged commands and anomalous outbound network connections. Indicators suggest credential misuse consistent with MITRE T1078.

Impact:

- Potential compromise of Server-Y and access to sensitive services.

- Possible credential exposure and lateral movement risk to internal systems.

Immediate Actions Taken:

- Isolated Server-Y from the production network (removed from VLAN / applied ACL).

- Terminated active sessions and disabled externally facing services where appropriate.

- Created TheHive case and escalated to Tier 2. (Case ID: [insert])

- Preserved volatile evidence: auth logs, running processes, memory snapshot (if captured), and packet captures for the incident time window.

Evidence Collected:

- /var/log/auth.log extracts (timestamps centered on 2025-08-18 12:55–13:15)

- Process snapshot: ps aux output

- Network flows / pcap: tcpdump capture (saved as evidence\_2025-08-18.pcap)

- Alerts and correlation artifacts from Wazuh / Elastic Security

Preliminary Findings:

- Authentication from a remote vector led to an account being used to run a sequence of sudo-level commands.

- Outbound DNS and HTTP connections to unknown external hosts were observed immediately following the activity.

Recommended Next Steps (Tier 2):

1. Full forensic triage of Server-Y (disk image, memory analysis).

2. Validate scope: scan for lateral movement and reuse of credentials on other hosts (focus on 192.168.204.138).

3. Force password resets for impacted accounts and require MFA for privileged logins.

4. Block IOCs at perimeter and endpoint controls; add IOCs to Wazuh/OTX lists.

5. Rebuild or reimage Server-Y if persistence/malware is confirmed.

Communications:

- Notified IT Ops and InfoSec lead at detection time T+10m.

- Recommended messaging drafted for impacted business owners.

Points of Contact:

- SOC Tier 1 Lead: [Name / phone / email]

- Tier 2 Forensics Lead: [Name / phone / email]

- IT Ops: [Name / phone / email]

Appendix:

- How to reproduce the alert (lab): a hydra-based SSH brute force simulation was run against 192.168.204.137 to validate detection tuning.

- Location of artifacts: /opt/evidence/incidents/2025-08-18-ServerY/

**4) Simulate the Attack (Hydra) — commands (lab use only)**

**Warning / legal note:** Only run these tests against systems you own or have explicit permission to test.

Simulate SSH brute force to generate noisy auth events and exercise detection.

sudo apt update && sudo apt install -y hydra

# Simulate a credential attack against Server-Y (Debian 192.168.204.137)

hydra -l admin -P /usr/share/wordlists/rockyou.txt -t 4 ssh://192.168.204.137 -s 22

Collect supporting evidence on Server-Y as soon as possible:

# Collect auth logs

sudo cp /var/log/auth.log /opt/evidence/ServerY\_auth\_2025-08-18.log

# Capture 60 seconds of network traffic around the event

sudo tcpdump -n -i eth0 -w /opt/evidence/ServerY\_2025-08-18.pcap -G 60 -W 1

# Snapshot running processes

ps aux > /opt/evidence/ServerY\_ps\_2025-08-18.txt

# Hash evidence files for chain of custody

sha256sum /opt/evidence/\* > /opt/evidence/manifest.sha256

**5) Splunk Phantom — Simple Playbook to Auto-Assign High Alerts to Tier 2**

Below is a **straightforward playbook outline** you can implement in Phantom (Drag-drop GUI or via REST). It checks incoming alerts, and when priority=High it assigns the case to the "Tier 2" group and creates a TheHive case.

**5.1 Playbook steps (logical flow)**

1. **Trigger:** Alert received (mock or real) in Phantom (artifact contains priority, alert\_id, source\_ip, target\_ip, description).
2. **Condition:** If priority == "High" (or numeric >= 3) → proceed; else finish (route to Normal queue).
3. **Action 1:** Add note to Phantom artifact: "Auto-escalate to Tier 2".
4. **Action 2:** Set owner / container field to Tier 2 (assign in Phantom).
5. **Action 3:** Create a TheHive case via TheHive API with the alert metadata (use HTTP connector).
6. **Action 4:** Post notification to Slack/email to #soc-escalations with quick summary and TheHive case link.
7. **Action 5:** Close or mark artifact as escalated.

**5.2 Example Phantom pseudo-code (Python snippet using Phantom REST actions)**

In Phantom you normally use the UI connectors; below is a conceptual Python code / sample that demonstrates the logic using Phantom’s phantom helper methods inside a custom function block.

def on\_start(container):

alert = container.get('data', {})

priority = alert.get('priority', 'Low')

if priority.lower() == 'high' or int(alert.get('severity',0)) >= 3:

phantom.add\_note(container=container, note\_type="general", title="Auto-escalation", content="Auto-escalate to Tier 2")

# Assign owner/team in Phantom

phantom.set\_owner(container=container, user="tier2\_team")

# Create case in TheHive

thehive\_payload = {

"title": f"Auto Escalation - Alert {alert.get('alert\_id')}",

"description": alert.get('description'),

"severity": 3,

"tlp": 2,

"tags": ["auto-escalated","Tier2"]

}

phantom.act("http post", parameters={"url":"https://THEHIVE\_URL/api/case","headers":{"Authorization":"Bearer YOUR\_THEHIVE\_APIKEY","Content-Type":"application/json"},"data":json.dumps(thehive\_payload)})

# Notify Slack/email (use connector)

phantom.act("send message to slack", parameters={"channel":"#soc-escalations","message":f"High alert auto-assigned to Tier 2: {alert.get('alert\_id')}. TheHive case created."})

else:

phantom.add\_note(container=container, note\_type="general", title="No automation", content="Alert routed to standard queue")

phantom.finish(container=container)

*(Adapt to the GUI by wiring connectors: HTTP connector → TheHive, Slack or Email connector for notifications, and a “set owner” action.)*

**5.3 Mock alert to test playbook (sample artifact)**

Drop this JSON as a test artifact in Phantom (or use the “Create Event” test in Phantom UI):

{

"alert\_id": "test-0001",

"priority": "High",

"severity": 4,

"source\_ip": "203.0.113.55",

"target\_ip": "192.168.204.137",

"description": "Unauthorized SSH login observed on Server-Y; repeated sudo activity; potential valid account compromise (T1078)."

}

**Expected results after playbook run:**

* Phantom container owner becomes Tier 2.
* TheHive case created (check TheHive UI or API).
* Slack/email notification posted to #soc-escalations.
* Phantom container annotated with escalation notes and link to TheHive.