**Priority Definitions**

* **Understand severity levels** such as **Critical, High, Medium, Low** — typically defined by:  
    
  + **Impact** (e.g., data breach, service disruption)
  + **Urgency** (e.g., active exploitation)
  + **Example mapping**:  
      
    - **Critical**: ransomware encryption, immediate business shutdown
    - **High**: unauthorized administrative access
    - **Medium/Low** follow similar logic based on business and technical evaluation

**2. Assignment Criteria — Alert Prioritization**

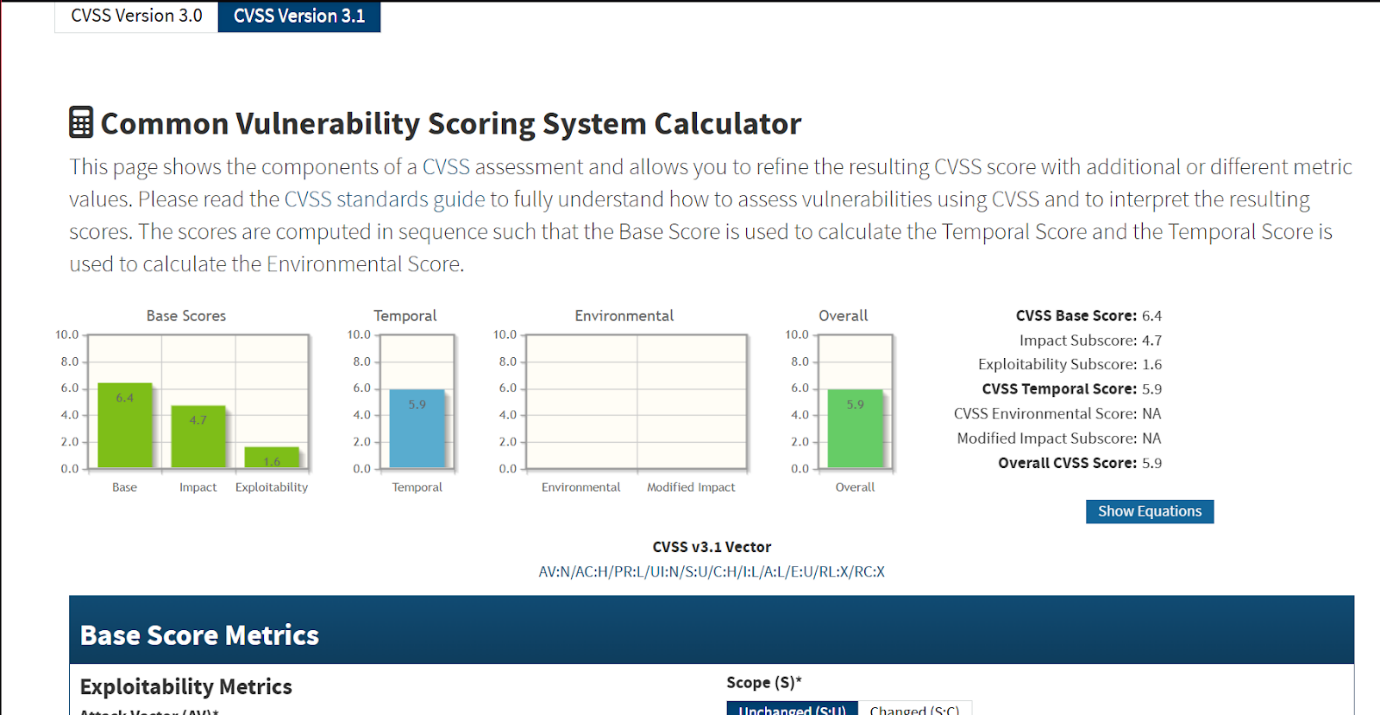
When ranking alerts, consider:

* **Asset Criticality**: production vs. test systems
* **Exploit Likelihood**: known CVE, public proof-of-concept
* **Business Impact**: financial loss, compliance exposure
* **Example**: A CVSS score of 9.8 (e.g., Log4Shell, CVE-2021-44228) often maps to a **Critical** priority level

**3. Scoring Systems and Risk Quantification**

SOC teams don’t just rely on gut feelings—they use **standard scoring systems** to reduce bias.

* **CVSS (Common Vulnerability Scoring System)**:  
    
  + **Base Metrics** measure intrinsic severity (constant traits)
  + **Temporal Metrics** reflect time-sensitive factors (e.g., exploit maturity, patch availability)
  + **Environmental Metrics** consider your environment’s specifics and business context
  + In **CVSS v4.0**, there are four groups—Base, Threat (similar to Temporal), Environmental, and Supplemental [FIRST](https://www.first.org/cvss/user-guide?utm_source=chatgpt.com)[Wikipedia](https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System?utm_source=chatgpt.com).
  + The **National Vulnerability Database (NVD)** provides Base scores and a CVSS calculator—but not full Temporal or Environmental assessments [NVD](https://nvd.nist.gov/vuln-metrics/cvss?utm_source=chatgpt.com).



**SOC Risk Scoring (SIEM-based)**

* Tools like Splunk or Elastic assign **risk points** based on:  
    
  + The type of event (e.g., malware detection = +50).
  + The asset affected (e.g., production server = +30).
  + The user involved (e.g., admin account = +20).
* When risk exceeds a threshold (say 80+), the alert is **Critical**.

* **Severity categories** (common for v3.x and v4.0):  
    
  + **Low**: 0.1–3.9
  + **Medium**: 4.0–6.9
  + **High**: 7.0–8.9
  + **Critical**: 9.0–10.0 [NVD](https://nvd.nist.gov/vuln-metrics/cvss?utm_source=chatgpt.com)[Cobalt](https://www.cobalt.io/blog/understanding-the-cvss-base-score-an-essential-guide?utm_source=chatgpt.com)[Wikipedia](https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System?utm_source=chatgpt.com)
* **Temporal metrics** example: availability of proof-of-concept or patch can lower or raise overall score [Information Security Stack Exchange](https://security.stackexchange.com/questions/256741/cvss-temporal-guidance?utm_source=chatgpt.com)[Wikipedia](https://en.wikipedia.org/wiki/Common_Vulnerability_Scoring_System?utm_source=chatgpt.com).
* **Environmental example**: a banking web server vulnerability may be scored higher if critical to service and widely deployed [NIST](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198&utm_source=chatgpt.com).

**4. NIST Incident Severity Classification & Prioritization Workflows**

* **NIST SP 800-61 Revision 3 (released April 2025)** is now the **latest**, replacing the archived Rev 2 [NIST Publications+1](https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf?utm_source=chatgpt.com).
* It outlines **incident prioritization**, **severity estimation**, **response initiation**, and **recovery processes** [NIST Publications](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf?utm_source=chatgpt.com).
* NIST emphasizes using **functional impact**, **information impact**, and **recoverability** to prioritize — particularly for third-party or supply-chain incidents [Mitratech](https://mitratech.com/resource-hub/blog/nist-sp-800-61/?utm_source=chatgpt.com).
* The older **SP 800-61 Revision 2** (from 2012, now withdrawn) still provides foundational guidance on incident analysis and response—but should be referenced for historical context only [NIST Computer Security Resource Center](https://csrc.nist.gov/pubs/sp/800/61/r2/final?utm_source=chatgpt.com)[NIST Publications](https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-61r2.pdf?utm_source=chatgpt.com).
* **Practical frameworks** based on these NIST standards—for example, incident response playbooks and templates—outline the four lifecycle phases: Preparation; Detection & Analysis; Containment, Eradication & Recovery; and Post-Incident Activity [Concertium](https://concertium.com/nist-incident-response-playbook-template/?utm_source=chatgpt.com)[NIST Computer Security Resource Center](https://csrc.nist.gov/pubs/sp/800/61/r2/final?utm_source=chatgpt.com)[NIST Publications](https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r3.pdf?utm_source=chatgpt.com).

**5. Key Learnings from Study**

Through my study of alert prioritization frameworks and scoring systems, I gained several important insights:

1. **CVSS Scoring**
   * I learned how **CVSS (Common Vulnerability Scoring System)** provides a structured way to measure the severity of vulnerabilities.
   * The **Base metrics** give a static severity assessment (e.g., impact on confidentiality, integrity, availability).
   * **Temporal metrics** reflect real-world exploitability factors like whether a proof-of-concept exploit is publicly available.
   * **Environmental metrics** allow organizations to adjust severity based on their specific context, such as the criticality of the affected asset or compensating controls.
   * I also saw how **severity bands** are categorized: Low (0.1–3.9), Medium (4.0–6.9), High (7.0–8.9), and Critical (9.0–10.0). This mapping helps translate CVSS scores into alert priority levels.
2. **Incident Prioritization Workflows (NIST Guidance)**
   * From the **NIST incident handling guide (SP 800-61)**, I learned that prioritization doesn’t rely on technical severity alone — it also considers **functional impact**, **information impact**, and **recoverability**.
   * For example, an attack with limited technical severity could still be treated as high-priority if it affects regulated data or damages business reputation.
   * NIST also emphasizes a lifecycle approach: **Preparation; Detection & Analysis; Containment, Eradication & Recovery; and Post-Incident Activity**. Prioritization is embedded in the **Detection & Analysis** phase, guiding how quickly the SOC responds.
3. **Practical Application**
   * A case study like **Log4Shell (CVE-2021-44228)** helped me connect theory to practice. Its CVSS score of 9.8 clearly placed it in the **Critical** category, but what made it urgent was not only the score, but also its widespread exploitability and business impact (threats to production systems worldwide).
   * This reinforced that **scores are a starting point**, but SOC analysts must also weigh exploit availability, asset exposure, and organizational context before finalizing alert priority.
4. **Takeaways for SOC Operations**
   * Alert prioritization is essentially about balancing **quantitative scoring** (like CVSS) with **qualitative judgment** (like business impact).
   * By combining these, analysts can reduce noise, focus on the most pressing threats, and align response efforts with organizational risk tolerance.

2. Incident Classification

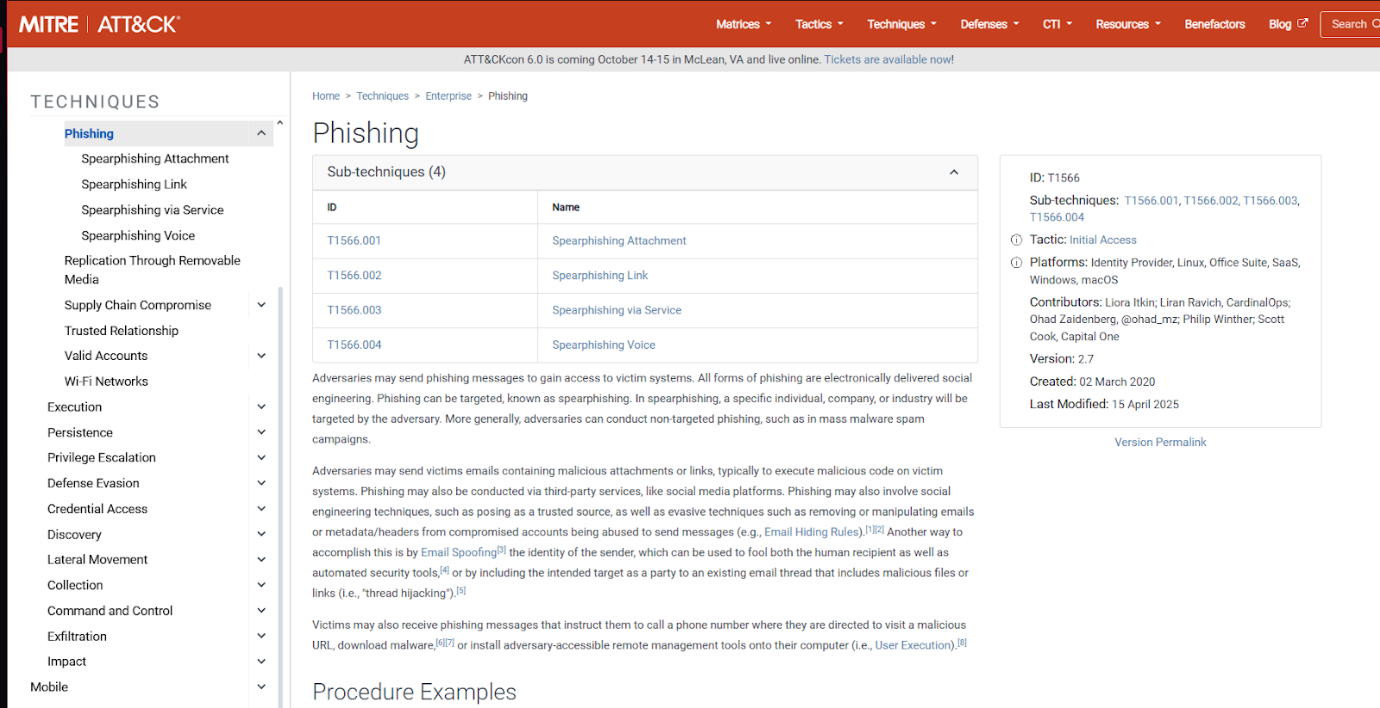
**Core Concepts:**

Through my study of incident classification frameworks and case studies, I developed a strong understanding of how incidents are systematically categorized and enriched to support investigations.

1. **Incident Categories**
   * I learned that incidents are commonly classified by type, such as **malware infections, phishing attacks, denial-of-service (DoS/DDoS), insider threats, and data exfiltration**.
   * Each category carries distinct response considerations. For example, **insider threats** may involve unauthorized data exports by employees or contractors, requiring both technical and HR/legal response processes.
   * Recognizing these categories allows analysts to quickly align alerts with appropriate playbooks and escalation paths.
2. **Taxonomies and Frameworks**
   * I studied several frameworks that provide **standardized language and structure** for incident classification:  
       
     + **MITRE ATT&CK**: maps incidents to specific tactics and techniques (e.g., *T1566 – Phishing* under the Initial Access tactic).
     + **ENISA Incident Taxonomy**: defines high-level categories for consistent reporting across organizations in Europe.
     + **VERIS (Vocabulary for Event Recording and Incident Sharing)**: focuses on structured incident reporting, especially for sharing data across organizations.
   * By applying these taxonomies, incidents can be labeled consistently, which helps with **trend analysis, reporting, and threat intelligence sharing**.
3. **Contextual Metadata Enrichment**
   * Beyond categorization, I learned the importance of enriching incidents with **contextual metadata** such as affected assets, system roles (e.g., production vs. test), timestamps, attacker IPs, and Indicators of Compromise (IOCs) like file hashes or malicious domains.
   * This enrichment transforms a raw alert into a **complete incident record**, making it easier for responders to analyze patterns and determine scope.

In addition to understanding the theory of incident categories, I explored several frameworks and real-world examples to build practical skills in classification and enrichment:

1. **MITRE ATT&CK Framework**
   * I studied how **MITRE ATT&CK** breaks down adversary behavior into **tactics (the “why”)** and **techniques (the “how”)**.
   * For example, phishing (T1566) falls under the **Initial Access** tactic, but ATT&CK further breaks it down into **spearphishing attachment**, **spearphishing link**, and **spearphishing via service**.
   * Mapping incidents to ATT&CK helped me move beyond just saying “phishing happened” to clearly identifying the attacker’s method and intent.
   * I also learned how ATT&CK **matrices and detection mappings** can be used in a SOC to standardize how incidents are documented and investigated.



1. **ENISA Incident Taxonomy**
   * From studying ENISA’s work, I learned that their taxonomy provides **broad, high-level categories** like Malicious Code, Unauthorized Access, Information Leakage, and Availability.
   * What stood out was ENISA’s emphasis on **uniform reporting across organizations and countries**. This showed me the importance of classification not just for internal investigations but also for compliance and cross-border information sharing.
2. **VERIS Framework (Vocabulary for Event Recording and Incident Sharing)**
   * I studied VERIS as a structured way to document incidents for **metrics and analysis**.
   * Unlike ATT&CK, which focuses on attacker behavior, VERIS emphasizes **who, what, why, and how** of an incident.
   * For example, in a phishing case, VERIS captures not just that phishing occurred, but also details such as **actor type (external), action (social engineering), asset (email server, endpoint), and impact (credentials compromised)**.
   * I realized that VERIS is especially powerful for **data-driven analysis**, like the Verizon DBIR (Data Breach Investigations Report), which is built on VERIS.
3. **Case Studies and Practical Exercises**
   * I reviewed phishing campaign case studies (such as those in the SANS Reading Room). These showed how analysts apply classification and metadata enrichment in real investigations.
   * For instance, one phishing campaign was mapped as:  
       
     + **Category**: Phishing (ENISA)
     + **Technique**: T1566.001 – Spearphishing Attachment (MITRE ATT&CK)
     + **Metadata**: Malicious macro-enabled document (MD5 hash provided), delivery timestamp, source IP ranges, and affected user accounts
     + **VERIS Recording**: Actor = External, Action = Social, Asset = User email account, Impact = Confidentiality breach (credential loss)
   * This exercise made it clear how these frameworks complement each other: **ATT&CK** for behavior, **ENISA** for broad reporting, **VERIS** for structured incident sharing.

3. Basic Incident Response

**Core Concepts:**

While studying incident response, I focused on both the theoretical lifecycle and the practical skills required to manage real-world incidents. The main insights I gained are:

**1. Incident Lifecycle**

* I learned that the **incident response lifecycle** follows six key phases, as defined by NIST and reinforced by SANS:  
    
  1. **Preparation** – building readiness through playbooks, policies, logging strategies, and tools. For example, having a phishing playbook ensures the SOC responds consistently every time.
  2. **Identification** – triaging alerts to confirm whether an actual incident is occurring. This involves reviewing logs, IOCs, and threat intelligence to distinguish between false positives and real threats.
  3. **Containment** – limiting the spread of an incident. I studied examples like isolating compromised endpoints from the network or disabling a malicious user account.
  4. **Eradication** – removing the root cause of the attack, such as deleting malware, closing exploited vulnerabilities, or resetting compromised credentials.
  5. **Recovery** – restoring systems to production, often in stages. For example, bringing a patched web server back online gradually while monitoring for reinfection.
  6. **Lessons Learned** – conducting a **post-mortem** to analyze what worked, what failed, and how processes or defenses can be improved.

**2. Core Procedures**

* I studied the **practical procedures** that SOC teams apply during incidents:  
    
  + **System isolation**: Using EDR tools or firewall rules to quarantine compromised hosts.
  + **Evidence preservation**: Capturing volatile data (e.g., memory dumps, running processes), creating forensic images, and hashing files to maintain chain of custody.
  + **Communication protocols**: Following escalation paths, informing stakeholders in non-technical language, and ensuring no sensitive information leaks outside the approved channels.
  + **SOAR (Security Orchestration, Automation, and Response)**: I explored how tools like **Splunk Phantom** or **Cortex XSOAR** automate repetitive tasks such as blocking IPs or generating incident tickets, freeing analysts to focus on higher-level analysis.

As I studied incident response, I approached it from three perspectives: **frameworks (NIST), best practices (SANS), and hands-on simulation (Let’s Defend and SOAR tools).**

**1. NIST SP 800-61 Incident Handling Guide**

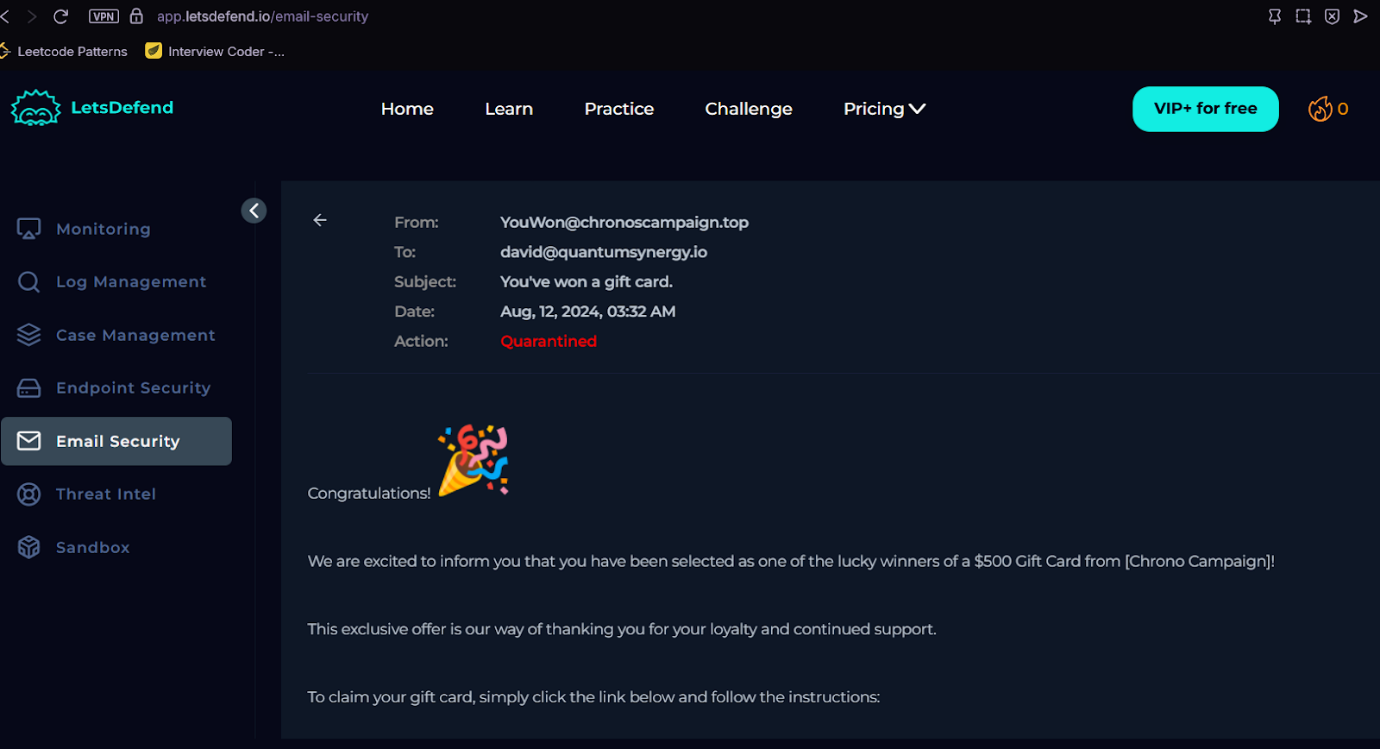
* I studied the NIST guide in depth and found it invaluable for understanding the **structured lifecycle of incident response**.
* It emphasized that the most important phase is **Preparation**, since the effectiveness of response depends on readiness (playbooks, trained staff, logging, detection systems).
* I learned that NIST categorizes incidents based on **functional impact**, **information impact**, and **recoverability**, which influences how the lifecycle phases are executed.
* A key insight I gained is that **incident response is iterative** — for example, containment strategies may be short-term (quarantining a system) or long-term (deploying permanent firewall rules), and these decisions are made dynamically.
* Studying NIST also showed me how important **documentation and evidence tracking** are for both compliance and future lessons learned.

**2. SANS Incident Handler’s Handbook**

* The SANS handbook translated NIST’s lifecycle into **practical, analyst-friendly checklists and templates**.
* I learned how to perform actions such as:  
    
  + Writing an **incident diary** to capture evidence and decisions in real time.
  + Following a **containment checklist** (e.g., disconnecting a host, revoking credentials, or disabling services).
  + Conducting **post-incident reviews** with both technical teams and business stakeholders.
* The handbook also emphasized **communication protocols**, which was something I hadn’t initially considered as deeply technical, but it’s actually critical to response success. For example, during containment, SOC analysts should notify business owners before shutting down a production system to avoid unnecessary downtime.
* Overall, the SANS perspective helped me see how to bridge the gap between **theory and real SOC operations**.

**3. Hands-On Practice (Let’s Defend and SOAR Concepts)**

* I applied what I studied through **Let’s Defend labs**, which provided **simulated SOC scenarios**. These exercises gave me confidence in moving through the lifecycle phases, especially **Identification** (triaging alerts and recognizing true positives) and **Containment** (isolating compromised hosts).  
  
* In one phishing simulation, I practiced gathering indicators from malicious emails, analyzing them, and correlating them with MITRE ATT&CK techniques — reinforcing how incident classification ties into response.



* I also explored **SOAR (Security Orchestration, Automation, and Response)** tools like Splunk Phantom. I learned how automation can:  
    
  + Automatically enrich alerts with threat intelligence.
  + Trigger containment actions like blocking IPs or disabling accounts.
  + Generate standardized reports.

These simulations highlighted that **automation speeds up repetitive tasks**, but human judgment is still required for high-impact decisions, especially during eradication and recovery