SPLK-2003 Training Course

Splunk SOAR Certified Automation Developer Exam

Structured Learning & Certification Preparation

Table of Contents

Introduction

The SPLK-2003 Splunk SOAR Certified Automation Developer certification is intended to reflect practical understanding of how automation is designed, configured, and extended within a Splunk SOAR environment. It represents the ability to work with core platform functions, playbook development concepts, integrations, and operational customization. In a modern security operations context, this knowledge is valuable because organizations increasingly rely on orchestration and automation to improve consistency, reduce manual effort, and support faster response workflows.

About This Training / Certification

This certification is centered on the skills needed to understand and develop automation in Splunk SOAR while also working effectively with the surrounding platform capabilities that support investigations, case handling, and system operation. It is best viewed as an intermediate-level certification for learners who already have a basic understanding of security operations, platform administration concepts, and scripting or logical workflow design. In a broader learning journey, it fits between foundational product familiarity and more advanced, real-world automation engineering, helping candidates move from knowing how the platform works to understanding how to build useful, maintainable automated processes within it.

What We Offer (AAAdemy)

AAAdemy provides structured training resources designed to support certification preparation and skill development across a wide range of IT domains. Our learning materials are built around clear knowledge structures, practical study guidance, and exam-oriented practice to help learners progress with confidence.

We offer well-organized knowledge explanations that break down complex topics into clear, understandable sections aligned with official exam objectives and real-world skill requirements. Each topic is designed to support both conceptual understanding and practical application.

Our study plans and learning guidance help learners follow a logical progression, focusing on key concepts, common pitfalls, and effective preparation strategies. This approach enables learners to study efficiently while maintaining a clear view of their learning goals.

To reinforce understanding, AAAdemy also provides practice questions and exam-focused insights that reflect typical certification scenarios. These resources are intended to help learners evaluate their readiness and strengthen their confidence before taking an exam.

All content is designed for flexible, self-paced learning, allowing individuals to study independently or alongside their existing professional or academic commitments.

Knowledge Overview

Domain: Deployment, Installation, and Initial Configuration

This domain focuses on the foundational aspects of establishing a functional Splunk SOAR environment. Candidates are expected to understand the considerations involved in deploying the platform, including system readiness, configuration dependencies, and initial setup decisions that influence long-term usability. This includes awareness of how environment configuration impacts integration capabilities, system performance, and operational scalability. A conceptual understanding of how the platform transitions from installation to an operational security tool is essential.

Domain: User Management

This domain addresses how access to the platform is structured and controlled. Candidates should understand the principles of role-based access control, including how permissions are assigned and enforced to align with organizational responsibilities. This includes managing user roles, ensuring proper separation of duties, and maintaining security boundaries within the system. The emphasis is on enabling controlled collaboration while protecting critical automation capabilities.

Domain: Apps, Assets, and Playbooks

This domain explores the core building blocks of automation within Splunk SOAR. Candidates should understand how apps provide integration capabilities with external systems, how assets represent configured and authenticated connections, and how playbooks orchestrate these components into executable workflows. The focus is on understanding the relationships between these elements and how they collectively enable automated security operations.

Domain: Analyst Queue

This domain focuses on how incoming events and alerts are operationalized within the platform. Candidates should understand how the analyst queue serves as the entry point for investigative work, including how items are prioritized, categorized, and assigned. It also involves understanding how automation can augment triage processes and reduce manual workload by pre-processing events before analyst interaction.

Domain: The Investigation Page

This domain centers on the analytical workspace used during incident review. Candidates should understand how data related to an event is aggregated, visualized, and enriched to support decision-making. This includes awareness of how contextual information is presented and how automation contributes to building a more complete investigative picture.

Domain: Case Management and Workbooks

This domain focuses on structured incident response and workflow standardization. Candidates should understand how cases are used to track incidents over time and how workbooks provide predefined investigative steps to ensure consistency. The emphasis is on enabling repeatable, auditable processes that support collaboration and operational discipline.

Domain: Customizations

This domain involves adapting the platform to meet specific operational requirements. Candidates should understand how fields, layouts, and configurations can be modified to better reflect organizational workflows. The focus is on aligning the platform’s interface and behavior with real-world use cases without compromising maintainability.

Domain: System Maintenance

This domain addresses the ongoing management of the platform to ensure reliability and performance. Candidates should understand general maintenance practices, including monitoring system health, managing updates, and ensuring that integrations and automations continue to function as expected in a changing environment.

Domain: Introduction to Playbooks

This domain establishes the conceptual foundation of automation within Splunk SOAR. Candidates should understand what playbooks are, how they are structured, and how they execute. This includes recognizing the role of playbooks in orchestrating actions across multiple systems and supporting security workflows.

Domain: Visual Playbook Editor

This domain focuses on the graphical interface used to design automation workflows. Candidates should understand how to construct, organize, and manage playbook logic visually. This includes arranging actions, defining execution flow, and maintaining clarity in workflow design for ease of understanding and future modification.

Domain: Logic, Filters, and User Interaction

This domain emphasizes the control mechanisms within playbooks. Candidates should understand how conditional logic drives decision-making, how filters refine data inputs, and how user interaction points allow for controlled manual input. The focus is on designing workflows that can adapt dynamically based on data and context.

Domain: Interaction

This domain specifically focuses on how playbooks incorporate direct user engagement. Candidates should understand how prompts, approvals, and manual decision points are implemented within workflows. This enables automation to pause, gather input, and continue execution in a controlled and auditable manner.

Domain: Formatted Output and Data Access

This domain covers how playbooks retrieve, manipulate, and present data. Candidates should understand how data is accessed from various sources, transformed into usable formats, and delivered as structured output. The emphasis is on ensuring that automation results are meaningful and actionable.

Domain: Modular Playbook Development

This domain focuses on designing scalable and maintainable automation. Candidates should understand how to break down complex workflows into reusable modules, enabling consistency and reducing duplication. The goal is to support long-term maintainability and efficient development practices.

Domain: Custom Lists and Data Routing

This domain involves managing structured data within the platform and directing it through workflows. Candidates should understand how lists are used to store reusable data and how routing logic determines the flow of information between playbook components.

Domain: Configuring External Splunk Search

This domain addresses how external Splunk search capabilities are integrated into SOAR workflows. Candidates should understand how search queries are configured and used to retrieve relevant data that enhances automation and investigation processes.

Domain: Integrating SOAR into Splunk

This domain focuses on the interoperability between Splunk SOAR and the broader Splunk ecosystem. Candidates should understand how data flows between platforms and how integrations support a unified approach to security operations.

Domain: Custom Coding and Using REST

This domain emphasizes extending platform capabilities beyond built-in features. Candidates should understand how custom code and REST APIs are used to integrate with external systems, implement advanced logic, and support complex automation scenarios.

Detailed Knowledge Explanation

1. SPLK-2003 Analyst Queue

The Analyst Queue serves as the strategic "control room" for SOC operations within Splunk SOAR. By centralizing incident management into a single, real-time dashboard, it allows security teams to effectively manage the influx of security data, ensuring that every alert is accounted for and that analysts maintain clear accountability throughout the investigation lifecycle.

1. Feature Overview and Functional Utility

At its core, the Analyst Queue functions as a sophisticated task inbox specifically designed for security incidents, alerts, and cases. It transforms raw, ingested data into an organized workspace, allowing analysts to determine what actions are required, prioritize their efforts, and manage ownership of specific tasks.

Feature	Purpose
Prioritized Feed	Shows analysts the most important events first based on severity and tags.

Manual Assignment	Enables analysts to claim events, establishing clear ownership.
Auto Assignment	Uses rule-based logic to automatically route events to specific teams.
Filters	Helps analysts focus on specific event types, sources, or risk levels.

2. Core Operational Features

2.1 Prioritized Event Feed

The event feed contains "containers" (the technical term for events) and is updated in real time. It is strategically sorted by severity, tags (e.g., “malware”), and timestamps. This ensures analysts address the most urgent threats immediately rather than parsing raw datasets.

2.2 Manual Assignment Mechanisms

Splunk SOAR utilizes a manual "claim" workflow where events begin in an **Unassigned** state. Analysts can manually assign incidents to themselves or others. This process establishes clear accountability and prevents operational redundancy within the SOC.

2.3 Automated Assignment Logic

For high-volume environments, SOAR employs a rule-based engine located under **Administration > Ingestion Settings > Ingest Settings Rules**. Rules can assign events based on severity or specialized team roles—such as routing "Phishing" tags to a specialized response team—ensuring rapid, specialized response.

2.4 Queue Filtering and Customization

Filtering empowers analysts to narrow their focus to high-risk activities. While administrators can set global presets, users can create and "pin" customized views to optimize personal workflows and reduce context switching.

Architect’s Exam Insight: Both *New* and *In Progress* statuses allow for interaction, but the certification exam will likely look for the fact that assignment typically begins when an event is in the **New** state.

3. Strategic Benefits for SOC Management

The Analyst Queue provides three primary advantages:

- **Balanced Workload:** Distributes incidents to prevent analyst fatigue.
- **Increased Visibility:** Managers can track who is handling which event in real time.
- **Faster Triage:** Priority events surface at the top, leading to a measurable reduction in time from detection to resolution.

4. Event Lifecycle and Status Transitions

Containers progress through a defined lifecycle:

- **New:** Default ingested state; appears in "Unassigned" views.
- **In Progress:** Indicates active work, set manually or via playbook.
- **Closed/Resolved:** Indicates completion. These are archived and hidden from active views by default.

5. Assignment and Role-Based Access Control (RBAC)

Visibility is governed by RBAC. **Administrators** possess full authority to assign or reassign any event. **Analysts** typically have permissions to assign events to themselves, while restricted roles may only see containers tagged for their specific team.

6. Default and Custom Views

The system provides predefined views like "All Open" and "Unassigned." Analysts can create persistent, custom filtered views for specific threats like phishing, improving operational efficiency.

The Analyst Queue effectively streamlines the transition from raw ingestion to active, accountable investigation, serving as the foundation for SOC efficiency.

7. Analyst Queue Practice Question

Q1: What is the primary purpose of the Analyst Queue in Splunk SOAR?
A. To centralize and manage incoming security events for triage
B. To configure external integrations
C. To display archived alerts for compliance audits
D. To create new roles and users for playbook management

Q2: When an analyst clicks “Assign to me” on an event in the Analyst Queue, what is the primary benefit?
A. It reduces the severity of the event
B. It increases the system timeout
C. It automatically runs the default playbook
D. It ensures clear ownership and prevents duplicate work

Q3: Which of the following best describes the function of Auto Assignment Rules?
A. They delete low-severity events from the queue
B. They automatically route events to analysts or teams based on rules
C. They determine when a playbook can be run
D. They allow users to prioritize events manually

Q4: Where can you configure automatic assignment rules in Splunk SOAR?
A. Settings > System Logs
B. Administration > Users
C. Administration > Ingestion Settings > Ingest Settings Rules
D. Playbook Editor > Assignments Tab

Q5: Which filter configuration would best help a senior analyst review only unresolved, high-risk phishing events from the last 24 hours?
A. Source = Splunk, Status = Closed, Tag = Email
B. Status = Open, Tag = Phishing, Severity = High, Created = Last 24 Hours
C. Owner = Me, Severity = Low, Tag = Spam
D. Asset = VirusTotal, Assigned = No, Created = Last 7 Days

Q6: What is the effect of leaving an event in the “Unassigned” state in the Analyst Queue?
A. It is hidden from all users
B. It is deleted after a timeout
C. It remains available for manual or automatic assignment
D. It is automatically escalated to the next shift

Q7: What is one of the main advantages of using queue filters in the Analyst Queue?
A. They change the severity of all events
B. They permanently delete irrelevant containers
C. They allow system logs to be downloaded
D. They help analysts focus on relevant incidents

Q8: In the Analyst Queue, which of the following is not a common sorting or filtering factor?
A. Response Time
B. Severity
C. Source
D. Tags

Q9: What is a key benefit of using Auto Assignment for large security teams?
A. It lets analysts reassign cases back to SOAR
B. It disables manual triage
C. It ensures timely distribution of incidents based on rules
D. It removes old assets from the queue

Q10: Which of the following is most accurate about workload management in the Analyst Queue?
A. Manual and auto assignments help balance team workload
B. All events are assigned to a single global queue
C. Events can only be assigned by administrators
D. Filters are fixed and cannot be customized

---

2. SPLK-2003 Apps, Assets, and Playbooks

The effectiveness of Splunk SOAR is built upon the symbiotic relationship between Apps, Assets, and Playbooks. Together, these components form the structural pillars that transform manual security procedures into a cohesive, automated response framework.

1. Component Relationships and Hierarchy

The system operates on a clear functional hierarchy:

- **Apps:** The software "adapters."
- **Assets:** The "configured accounts" (instances of those apps).
- **Playbooks:** The "instructional logic" (automated workflows). An App must be transformed into an Asset by providing credentials and network settings before it can be utilized by a Playbook.

Component	Purpose	Key Function
App	Enables integration	Provides the actions (e.g., "block ip") SOAR can call.
Asset	Custom instance	Stores API keys and config for real-world connectivity.
Playbook	Automation logic	Orchestrates assets to respond to security events.

2. App Integration Framework

Apps function as communication adapters between SOAR and external tools (e.g., VirusTotal, AWS). They are installed via the built-in **App Store** or **Manual Upload** (.tgz files) and provide predefined actions that playbooks call to execute tasks.

3. Asset Configuration and Security

Assets are specific instances of Apps containing authentication details (API keys) and network settings (URLs). Administrators can create multiple assets for a single app to segregate environments, such as a "Splunk-Test" asset for development and a "Splunk-Prod" asset for production.

4. Playbook Design and Execution

Playbooks represent the "automation logic," composed of Actions, Logic Blocks, and Prompts. They can be built using the drag-and-drop **Visual Editor** for speed or **Python scripting** for advanced requirements.

5. Asset Management: Tagging and Groups

To manage large-scale environments, analysts use user-defined tags (e.g., "Firewall," "Dev") to improve searchability. While "asset groups" are not a standalone feature, assigning the same tag to multiple assets allows analysts to effectively simulate and manage asset groups.

6. Playbook Validation and Troubleshooting

SOAR provides a **Test Mode** for safe execution against sample containers. For deeper logic troubleshooting, analysts can use `phantom.debug()` within code blocks to log specific variables to the system's `playbook.log`.

Architect's Exam Insight: If a playbook fails at a specific step, the most effective troubleshooting method is to review the **debug logs and output** of that specific block within the Investigation Page.

7. App Categorization by Use Case

- **Security Intelligence:** Enriching indicators (e.g., VirusTotal, Recorded Future).
- **Network/Firewall:** Traffic control and blocking (e.g., Palo Alto, Cisco Firepower).
- **Cloud Service:** Managing infrastructure (e.g., AWS, Azure).
- **ITSM/Tooling:** Workflow and notifications (e.g., JIRA, Slack).

These three components unify to transform disparate security tools into a synchronized, automated response ecosystem.

8. Apps, Assets, and Playbooks Practice Question

Q1: In Splunk SOAR, what is the primary purpose of an App?
A. To define which user roles can execute playbooks
B. To connect SOAR to an external system and expose actions
C. To store logs from connected tools
D. To define access control for playbooks

Q2: What distinguishes an Asset from an App in Splunk SOAR?
A. An asset is a configured instance of an app with credentials and settings
B. An asset is a scheduled action derived from an app
C. An asset is a visual representation of API calls
D. An asset is a playbook template built from an app

Q3: Which of the following best describes when a playbook in SOAR can be triggered?
A. When a user logs in via SAML
B. When an asset is modified
C. When an event is ingested, manually started, or scheduled
D. Only when a new app is installed

Q4: What would happen if a playbook references an action, but the asset it depends on is not configured?
A. The playbook will skip the asset
B. The action will run with default parameters

C. The asset will be auto-created using global settings
D. The playbook will fail at that step

Q5: In Splunk SOAR, how can you run a playbook?
A. Only during container ingestion
B. Automatically, manually, or based on schedule
C. From within the app configuration menu only
D. Only via Python code blocks

Q6: Which component is responsible for storing the authentication details such as API keys or user credentials?
A. Prompt block
B. Playbook
C. App
D. Asset

Q7: What feature allows playbooks to use smaller, reusable logic blocks that can be invoked from other playbooks?
A. Output filtering
B. Visual blocks
C. Playbook linking
D. Sub-playbooks (modular playbooks)

Q8: What is a typical use for a Format block inside a SOAR playbook?
A. Structuring data for outputs or prompts
B. Defining action timeouts
C. Launching new assets
D. Assigning roles to assets

Q9: Which of the following is true about using multiple assets from the same app?
A. You can create multiple assets with different configurations for different environments
B. Assets from the same app must use identical settings
C. Only one asset per app is supported
D. Only the default asset is used during playbook execution

Q10: What is the purpose of the “Test Connectivity” button when configuring an asset?
A. To validate the asset against the playbook logic
B. To check that the asset can successfully communicate with the external system
C. To publish the playbook using the asset
D. To automatically link it with the associated app

---

3. SPLK-2003 Case Management and Workbooks

As security events mature, they evolve into cases, where Workbooks provide a standardized roadmap for investigation. This structure ensures that incident response is not only rapid but also compliant with organizational protocols.

1. Case Management Principles

1.1 Automated and Manual Case Creation

Cases can be auto-generated based on triggers like severity or manually converted by an analyst from an existing event. This ensures the system remains focused on critical investigations.

1.2 Case Metadata and Field Structure

Essential fields (Title, Severity, Status, Owner, Tags) facilitate high-speed organization. Metadata allows for efficient searching and prioritization across the entire SOC workload.

1.3 Case Linking and Incident Aggregation

SOAR allows analysts to link multiple events to a single case. This is vital for aggregating related data, such as multiple endpoints affected by a single malware outbreak, providing a unified threat view.

1.4 Case Templates and Standardization

Case Templates ensure consistent response structures for repeatable incidents like phishing by defining required fields and default workbooks.

2. Workbook Architecture

2.1 Stages and Task Hierarchies

Workbooks are divided into **Stages** (e.g., Triage, Containment) and **Tasks**. This guides analysts through complex protocols and ensures no steps are missed.

2.2 Integration of Manual and Automated Tasks

Workbooks utilize a hybrid approach: **Manual Tasks** require human intervention, while **Automated Tasks** trigger playbooks. This maintains both speed and human oversight.

2.3 Progress Tracking and Performance Metrics

Status tracking (Pending, In Progress, Completed) enables managers to identify bottlenecks in the investigation lifecycle and report on team efficiency.

2.4 Customization of Response Workflows

Organizations can build custom workbook templates to align with internal security policies and compliance requirements.

3. Case Lifecycle: Resolution vs. Closure

There is a critical technical distinction between these states:

- **Resolved:** Technical actions (e.g., threat neutralized) are complete, but the case is still open for final review or documentation.
- **Closed:** The case has received administrative approval, is finalized, and is officially archived.

Architect's Exam Insight: Expect questions regarding the finality of a case. Only the **Closed** status indicates that the incident is fully reviewed and no further actions are required.

4. Operational Logic of Workbooks and Playbooks

Workbooks define the workflow and guide the investigation; they do not execute automation directly. Instead, tasks within a workbook are configured to **trigger** playbooks.

The underlying platform infrastructure provides the stability required to support these complex workflows across the enterprise.

5. Case Management and Workbooks Practice Question

Q1: What is the primary purpose of a Case in Splunk SOAR?
A. To manage and track the full lifecycle of a specific security incident
B. To execute scheduled system maintenance playbooks
C. To serve as a container for external integrations
D. To store archived logs of SOAR platform usage

Q2: How can a new Case be created in Splunk SOAR?
A. Only through manual input from administrators
B. Automatically based on event conditions or manually by analysts
C. Only after all playbooks finish execution
D. Via external scripts exclusively

Q3: Which of the following is not a standard field within a Splunk SOAR case?
A. Owner
B. Tags
C. Severity
D. Alert Volume

Q4: Why would an analyst use Case Linking in Splunk SOAR?
A. To allow unrelated events to share the same workbook
B. To separate malware alerts into distinct cases
C. To combine multiple related events under one investigation
D. To delete duplicates across different containers

Q5: What do Case Templates help standardize?
A. Raw log parsing across data sources

B. Role-based access for playbook permissions
C. Field structures, default values, and attached workbooks
D. Cloud connector integrations

Q6: In a Workbook, what is the relationship between Stages and Tasks?
A. Tasks and stages are unrelated execution blocks
B. Stages contain groups of tasks representing phases of an investigation
C. Each task is executed across all stages by default
D. Stages are smaller units within tasks

Q7: What distinguishes a Manual Task from an Automated Task in a workbook?
A. Manual Tasks require analyst input, while Automated Tasks run playbooks
B. Manual Tasks are triggered by SOAR timers
C. Automated Tasks involve email collaboration
D. Automated Tasks can only be added by Admins

Q8: What functionality does progress tracking offer in Splunk SOAR Workbooks?
A. It adjusts workbook templates dynamically
B. It identifies unresolved cases based on task and stage status
C. It auto-assigns analysts based on previous workload
D. It runs case performance benchmarks

Q9: How can an organization customize its workbook process in SOAR?
A. By modifying system logs
B. By editing event source settings
C. By creating custom workbook templates with tailored tasks and playbook links
D. By adjusting firewall policies via REST API

Q10: Which of the following is an example of a workbook task?
A. Updating license settings
B. Querying user login history during the triage stage
C. Editing source configuration
D. Syncing event artifacts to the cloud

---

4. SPLK-2003 Deployment, Installation, and Initial Configuration

Selecting the appropriate deployment model and ensuring a robust initial configuration are critical steps for achieving long-term SOAR stability and scalability.

1. Deployment Model Analysis

- **Standalone:** UI, Engine, and Database on one server. Ideal for training/labs; lacks production redundancy.
- **Distributed:** Separation of UI, Engine, and Database. Designed for high-volume environments, providing component-level resilience.
- **Cloud-Based:** Hosted in AWS/Azure or Splunk-managed instances, offering flexibility with reduced hardware overhead.

2. Installation Methodologies

- **Virtual Appliance (OVA):** Pre-built image for VMware/VirtualBox. Fastest setup but limited flexibility.
- **Manual/Scripted Linux:** Manual control on clean servers. Splunk officially supports:
  - **RHEL:** 7.6 – 8.x
  - **CentOS:** 7.6 – 7.9
  - **Ubuntu:** 18.04 and 20.04 LTS

3. Post-Installation Configuration

- **Network:** Static IPs, hostnames, and DNS are mandatory for tool integration.
- **Temporal Accuracy:** NTP synchronization is critical. Without it, the **Timeline View** and audit logs lose integrity.
- **Security:** SSL certificates protect sensitive data and fulfill compliance requirements.

4. Infrastructure and Licensing Requirements

- **Resources:** Minimum 4 CPU cores and 16 GB RAM. Production environments often require **500 GB+ disk space** for data retention.
- **Licensing:** Evaluation licenses last 30 days. Commercial licenses are tied to system UUID/MAC addresses.

Architect's Exam Insight: If a license expires, the platform will restrict logins and, crucially, **prevent new event ingestion and playbook execution.**

5. Administrative Management Tools

5.1 CLI Controls and Log Analysis

The `phantomd` daemon is the core service managed via CLI. Understanding log locations is vital for troubleshooting:

Log File Path	Diagnostic Purpose
`/var/log/phantom/phantomd.log`	**Authentication and Web activity**; main system log.
`/var/log/phantom/playbook.log`	**Playbook execution** and action results.

`/var/log/phantom/api.log`	API-driven interactions.
`/var/log/phantom/connector_<app>.log`	Specific App communication/connectivity issues.

5.2 Disaster Recovery

The `phantom_backup.py` tool automates backups of playbooks, assets, and cases. Best practice dictates storing these backups off-server (e.g., NFS or S3).

Once the system is properly configured, analysts interact with the data through the primary **Investigation Page**.

6. Deployment, Installation, and Initial Configuration Practice Question

Q1: Which of the following deployment types is most appropriate for a security team of three analysts who want to test SOAR in a lab environment?
A. Distributed Deployment
B. Manual Linux Installation
C. Cloud Deployment
D. Standalone Deployment

Q2: What is a key disadvantage of a standalone deployment of Splunk SOAR?
A. Poor scalability and single point of failure
B. High cost of licensing
C. Complexity of setting up networking
D. It requires cloud-specific expertise

Q3: A company wants to host SOAR on AWS with high availability and automatic resource scaling. Which deployment model is most appropriate?
A. Manual Linux Installation
B. Cloud Deployment
C. Distributed Deployment
D. Standalone Deployment

Q4: Which method provides the quickest way to deploy Splunk SOAR for demonstration or training purposes using VMware?
A. Using REST API
B. Cloud Deployment
C. Manual Linux Installation
D. Using OVA Images

Q5: During initial configuration, which step is critical for ensuring accurate alert timelines and log correlation in SOAR?
A. Setting a static IP address
B. Uploading a license

C. Configuring system time and timezone
D. Installing Python dependencies

Q6: Which of the following is a disadvantage of using an OVA-based installation in a production environment?
A. It cannot be integrated with external Splunk searches
B. It lacks flexibility in system resource customization
C. It requires advanced Linux expertise
D. It does not support SSL certificates

Q7: What is the purpose of running the SOAR setup wizard via web interface after installation?
A. To apply Linux firewall settings
B. To initiate the default playbook
C. To install system dependencies
D. To configure admin account, SMTP, and proxy

Q8: A team wants full control over the installation process and plans to use SOAR in production. Which method is most appropriate?
A. Manual Linux Installation
B. Using automated scripts
C. Using OVA image
D. Cloud Deployment

Q9: What tool can be used on a Linux server to interactively set a static IP address and DNS settings?
A. curl
B. soarctl
C. nmtui
D. timedatectl

Q10: Which statement about installing SSL certificates on SOAR is correct?
A. SSL setup is mandatory before running the setup wizard.
B. A self-signed certificate is always required before production use.
C. Trusted CA certificates can be configured to encrypt web interface access.
D. SSL must be configured through a command-line-only interface.

---

5. SPLK-2003 The Investigation Page

The Investigation Page acts as the central mission control for incident enrichment and response, providing analysts with the tools necessary to analyze evidence and execute defensive actions.

1. Workspace Components and Visual Timeline

The **Timeline View** provides a chronological audit trail of the incident, recording automated actions, manual comments, and status changes for compliance and auditing.

2. Artifact Intelligence and Management

Artifacts (IPs, Hashes, URLs) are evidence data displayed in a structured table. Analysts can interact directly with these data points to trigger enrichment or containment actions.

3. Manual Intervention and Action Panels

The **Manual Actions Panel** provides flexibility when human judgment is required. Analysts can manually trigger specific asset actions or playbooks as an investigation evolves.

4. Collaborative Documentation: The Notes Section

Notes record hypotheses, decisions, and team communications, ensuring all findings are documented in a single, accessible location.

5. Advanced Artifact Handling: Pinning and Criticality

Analysts can **"pin"** key artifacts to the Investigation Overview or flag them as **"critical."** This visually highlights high-risk indicators for triage and team hand-offs.

Architect's Exam Insight: To make a suspicious IP address stand out in the investigation view for other team members, the analyst should either **pin it** or mark it as **critical**.

6. Operational Monitoring and Case Reporting

- **Real-Time Supervision:** Analysts monitor live block status (success/failure) and can pause or abort playbooks if necessary.
- **Evidence Export:** Analysts can generate **Case Summary Reports** and **Audit Trail Documents** to meet regulatory requirements like GDPR or SOC2.

Investigative integrity is maintained by governing which users can access and perform these actions.

7. The Investigation Page Practice Question

Q1: What is the primary function of the Investigation Page in Splunk SOAR?
A. To manage all information, actions, and notes related to a specific incident
B. To run scheduled scans on external assets
C. To install apps and configure integrations
D. To monitor system health metrics and license usage

Q2: Which of the following is displayed in the Timeline View on the Investigation Page?
A. Analyst performance metrics
B. Only high-severity events

C. A chronological history of all actions, notes, and status changes
D. Only playbook-triggered actions

Q3: What is the purpose of the Artifacts Panel?
A. To assign events to analysts
B. To display evidence like IPs, hashes, URLs, and allow actions on them
C. To track SOAR server logs
D. To list event correlation rules

Q4: Which of the following actions can analysts perform from the Manual Actions Panel?
A. Delete artifacts from raw logs
B. Edit asset configuration
C. Reboot the SOAR platform
D. Manually trigger actions or playbooks using selected assets

Q5: Why is the Notes Section critical during an investigation?
A. It encrypts sensitive case data
B. It triggers automatic escalation of cases
C. It enables collaboration, logs decisions, and documents findings
D. It configures network device alerts

Q6: What is a typical use case for interacting with artifacts in the Investigation Page?
A. Running a data ingestion pipeline
B. Performing enrichment using threat intelligence tools
C. Deploying agents to endpoints
D. Launching infrastructure backups

Q7: How does the Investigation Page support blended automation?
A. By syncing firewall logs with case history
B. By allowing real-time modification of ingestion rules
C. By letting analysts intervene in playbook execution to pause, resume, or provide input
D. By exporting artifacts to external databases

Q8: Which of the following would not typically appear in the Investigation Page's Timeline View?
A. Asset configuration changes unrelated to the event
B. Status changes
C. Playbook block results
D. Analyst-added notes

Q9: What action might an analyst take directly from an artifact listed in the Investigation Page?
A. Add new data sources to SOAR
B. Change the user's assigned shift
C. Modify ingestion priority
D. Trigger a playbook on that specific artifact

Q10: Why is the ability to take manual action during a playbook run valuable?
A. It improves asset discovery speed

B. It allows analysts to guide or override automation in dynamic scenarios
C. It disables playbook output logging
D. It enables live collaboration between admins

---

6. SPLK-2003 User Management

Effective User Management through Role-Based Access Control (RBAC) is essential for maintaining a secure and auditable SOC environment, ensuring that users have access only to the data and tools required for their roles.

1. Core User Classification

- **Administrators:** Full platform control; management of users and apps.
- **Analysts:** Operational users who investigate alerts and manage cases.
- **Automation:** Machine accounts for API integration; **restricted from UI access.**

2. Role-Based Access Control (RBAC) Architecture

SOAR uses a "Role = Job Description" model. Permissions are assigned to roles, and roles are assigned to users, enabling secure scaling.

Role	UI Access	Can Run Playbooks	Can Modify System
Administrator	Yes	Yes	Yes
Analyst	Yes	Yes	No
Automation	**No**	Yes (via API)	No

3. Authentication Frameworks

- **Local:** Suitable for small teams; credentials stored in SOAR.
- **LDAP/Active Directory:** Enterprise-wide central management.
- **SAML (SSO):** Seamless access via Okta/Azure AD; highly secure.
- **2FA (TOTP):** Adds a second layer (e.g., Google Authenticator) for local accounts.

4. Account Security and Governance

Administrators can configure lockout policies for failed local login attempts and enforce password complexity and expiration intervals to satisfy compliance mandates.

5. Auditing and Compliance Logging

The **Audit Logs** track "who did what, when," covering login attempts, playbook executions, and system changes. These logs are essential for verifying analyst behavior and system integrity.

Architect's Exam Insight: A common exam trap involves the **Automation role**. Always remember that while it can trigger playbooks via the API, it is explicitly **restricted from logging into the web interface.**

These management controls ensure the integrity and accountability of the entire Splunk SOAR ecosystem.

6. User Management Practice Question

Q1: In Splunk SOAR, what is the primary benefit of using Role-Based Access Control (RBAC)?
A. It allows two-factor authentication for each user.
B. It lets users change their own permissions.
C. It prevents users from creating custom roles.
D. It simplifies access management by assigning permissions to roles.

Q2: Which user type is typically responsible for installing apps and configuring SMTP settings in Splunk SOAR?
A. Analyst
B. Administrator
C. Viewer
D. Junior Analyst

Q3: What is a key advantage of integrating SOAR with LDAP or Active Directory?
A. It allows users to modify their authentication methods.
B. It eliminates the need to manage users directly in SOAR.
C. It disables the need for multi-factor authentication.
D. It enables local password resets.

Q4: You need a role that allows users to view cases and run playbooks, but not modify assets. What's the best approach?
A. Use the built-in Administrator role.
B. Disable asset management in global settings.
C. Create a custom role with limited permissions.
D. Use the default Viewer role.

Q5: Which statement about local authentication in Splunk SOAR is correct?
A. It is ideal for small environments or testing purposes.
B. It allows user accounts to be stored externally.
C. It supports SAML-based single sign-on.
D. It cannot be combined with 2FA.

Q6: What is one possible drawback of using SAML-based SSO in Splunk SOAR?
A. It provides weaker encryption than local login.
B. Misconfigurations may prevent users from logging in.
C. It doesn't support large user groups.
D. It cannot be used with identity providers like Azure AD.

Q7: An analyst user needs to escalate a case, run playbooks, and add investigation notes. What permission set should they have?
A. SAML group role
B. View-only access
C. Analyst role or similar custom role
D. Administrator-level access

Q8: What benefit does enabling Two-Factor Authentication (2FA) provide in SOAR?
A. It removes the need for role-based permissions.
B. It adds a second layer of login security.
C. It reduces the need for password expiration.
D. It prevents users from using local accounts.

Q9: Which of the following best describes how users are granted permissions in SOAR?
A. Each user group has its own authentication method.
B. All permissions are managed through LDAP only.
C. Permissions are assigned directly to each user.
D. Permissions are assigned to roles, which are then applied to users.

Q10: Which authentication method in Splunk SOAR offers the fastest and simplest initial setup?
A. OAuth
B. SAML-based SSO
C. LDAP Integration
D. Local Authentication

---

7. SPLK-2003 Customizations

Customization serves as the primary mechanism for aligning the Splunk SOAR platform with unique organizational workflows, ensuring that the technology adapts to the team rather than forcing adherence to a rigid, pre-defined system. Architects must recognize that tailoring interfaces and integrations is not merely a cosmetic exercise but a strategic requirement for SOC efficiency. By refining how data is captured, visualized, and ingested, organizations transform the platform into a bespoke environment that mirrors their specific operational requirements and accelerates incident response.

1. Interface Customizations

The interface provides the direct point of contact between the security analyst and the automation platform. Modifying the analyst experience involves three primary methods designed to reduce cognitive load and enhance the precision of investigations. From an architectural perspective, these modifications ensure that the most critical data is surfaced during high-pressure incidents, allowing analysts to maintain situational awareness without navigating through extraneous noise.

1.1 Custom Fields

1. Custom fields are the linchpin for sophisticated SOC reporting and long-term trend analysis. While they allow for the capture of metadata unique to a business unit—such as "Region" or "Business Unit"—their strategic value lies in providing the structured data necessary for advanced search, filtering, and cross-case correlation. These fields are applied to three specific entities: Events (containers), Artifacts, and Cases.
2. Technical instructors must emphasize that custom fields are created and managed entirely through the Web UI by navigating to Administration > Custom Fields. This GUI-driven approach eliminates the need for Python code or YAML scripting. Available field types include text, dropdowns, checkboxes, and numbers, all of which can be configured as required or optional to ensure data integrity during ingestion.

1.2 Custom Layouts

1. The functional utility of custom layouts lies in the ability to reorganize the visual workspace to meet the specific needs of different user roles or case types. By rearranging panels on event and case pages, administrators can group related fields together or establish visibility rules that hide unnecessary data. This tailored environment is critical during high-pressure investigations, as it minimizes distractions and ensures that the analyst's focus remains on the most pertinent evidence.

1.3 Themes and Branding

1. For organizations that require a platform reflecting their corporate identity, the SOAR interface supports modification of visual elements such as logos, color schemes, and login page text. This level of customization is particularly vital for Managed Security Service Providers (MSSPs) who use SOAR as a component of a broader security suite provided to external clients, ensuring a professional and cohesive user experience that aligns with the provider's brand.

2. Automation Customizations

Automation customizations transform rigid, out-of-the-box behaviors into context-aware workflows through the use of custom actions, dynamic menus, and tagging logic. By moving beyond static automation, these features allow the platform to perform advanced logic and interface with internal tools intelligently. This shift reduces human error by ensuring that the data provided to analysts is valid and that the response paths taken by playbooks are precisely aligned with the nature of the threat.

2.1 Custom Actions

1. When standard app actions are insufficient, Python-based custom actions allow developers to extend the platform's capabilities. These actions can be integrated into apps or playbooks to perform advanced logic, execute API calls to proprietary internal tools, or conduct complex data transformations. This allows for specialized operations, such as querying an internal vulnerability scanner before a containment decision, to be seamlessly woven into the broader automation strategy.

2.2 Dynamic Menus

1. Dynamic menus replace static lists in user prompts by populating options at runtime based on live data. Supported data sources include internal SOAR objects such as Assets, User Groups, Roles, and Labels, as well as external system calls via custom REST API queries. A common misunderstanding among junior engineers is that only hardcoded values are permitted; however, dynamic menus pull live, valid data, which significantly increases the precision of analyst input and ensures selections are made from current system states.

2.3 Event Tags and Labels

1. Tags serve as a critical classification mechanism that facilitates both organization and automation triggering. By applying labels like "phishing," "malware," or "VIP user," playbooks can be configured to follow specific response paths automatically. For instance, an event tagged as "ransomware" can immediately trigger a high-priority containment playbook. These tags effectively serve as shortcuts to sophisticated, event-driven automation.

3. Integration Points

The integration architecture of Splunk SOAR supports both active and passive methods for data exchange. Architects must distinguish between these methods to ensure optimal performance. Active integration involves the platform initiating a request, whereas passive integration involves the platform listening for external signals.

3.1 Customize REST Endpoints

1. The SOAR REST API is the primary method for active interaction, where SOAR "asks or tells" something to another system. These endpoints can be extended to create new routes or modify existing ones to expose internal data and workflows. This is essential for teams developing custom dashboards or synchronization scripts that need to pull asset configuration details or push case updates in a request-driven manner.

3.2 Add Webhooks for External Alerts

1. Webhooks provide a passive method for real-time alert ingestion, where the "SOAR listens" for incoming data from external systems like firewalls and SIEMs. Instead of SOAR polling an external tool, the external system sends data directly to SOAR when an event occurs. This ingestion method is prized for its speed and simplicity, enabling the immediate creation of cases and the triggering of playbooks without the overhead of a full application integration.
2. These customization methods collectively create a flexible automation ecosystem that adapts to the unique threat landscape of any organization. Mastery of these interfaces and integration points is a prerequisite for moving into the mechanics of formatted output.

4. Customizations Practice Question

Q1: What is the purpose of adding custom fields in Splunk SOAR?
A. To track additional, organization-specific data within events, cases, or artifacts
B. To define access control roles
C. To override built-in system fields like severity
D. To delete system-generated tags

Q2: Which UI customization allows you to group fields visually and control their layout on case and event pages?
A. Tags
B. Custom layouts
C. Workbook stages
D. Playbook templates

Q3: In which scenario would a dynamic menu be most useful within a playbook prompt?
A. When offering analysts a static list of remediation steps
B. When automatically sending data to an external SIEM
C. When allowing analysts to choose from current asset lists pulled live
D. When adjusting firewall rules without user interaction

Q4: What is the primary purpose of custom actions in Splunk SOAR?
A. To adjust user authentication policies
B. To enrich events with built-in connectors only
C. To schedule data purging tasks
D. To extend automation by allowing Python-based actions within apps or playbooks

Q5: Which of the following best describes event tags in SOAR?
A. Fields that replace the workbook interface
B. System-generated alerts based on DNS rules
C. Labels used to classify events and optionally trigger playbooks
D. Indicators stored externally in Splunk Enterprise

Q6: A company wants to embed its branding into the SOAR interface. Which customization should they apply?
A. Change the theme, logo, and login page message
B. Modify REST endpoint access
C. Create a custom Python action
D. Edit default user roles and permissions

Q7: What is the role of webhooks in SOAR customization?
A. They perform periodic backups of case data
B. They allow Splunk SOAR to receive alerts directly from external tools in real time
C. They enforce session timeout settings
D. They replace the SOAR App Store

Q8: What does the REST API extension capability in SOAR enable?
A. Encrypting SOAR backups via API
B. Replacing system-wide playbook permissions

C. Disabling external alert ingestion from unsupported sources
D. Creating new API endpoints for custom integrations

Q9: How can tags be applied to events in SOAR?
A. Only manually through the case interface
B. Only through ingestion rules
C. Manually by analysts or automatically through playbooks and rules
D. Tags are not supported for security events

Q10: A security team wants to allow analysts to choose which internal tool to use during incident review. The tools change weekly. Which feature should be used?
A. Custom action
B. Static dropdown menu
C. Webhook configuration
D. Dynamic menu in a prompt block

---

8. SPLK-2003 Formatted Output and Data Access

In Splunk SOAR, clear communication of automated results is as critical as the execution of the automation itself. As playbooks process raw data—such as IP reputation scores or user account statuses—the transition from raw, technical output to actionable intelligence is achieved through formatting. Effective documentation ensures that both human analysts and downstream systems can interpret the outcomes of automated workflows without navigating complex JSON data structures.

1. Formatted Output

The Format Block is a specialized playbook component designed to synthesize raw data into structured, multi-line messages. By converting messy JSON or disparate variables into human-readable summaries, the platform ensures that the findings of a playbook are immediately understandable. Strategically, this allows the SOC to deliver consistent, professional-grade communication across various output channels.

1.1 Format Blocks and Message Templates

1. Format Blocks utilize text templates to combine static text with dynamic variables. The syntax allows for precise data targeting, such as `{artifact:*.cef.destinationAddress}` to extract a destination IP. These blocks support plain text, HTML for dashboard and email rendering, and limited Markdown for platforms like Slack or Microsoft Teams. It is a critical technical distinction that Format Blocks do not output raw JSON directly; such data must be preprocessed and converted to a string format via a code block.

1.2 Output Channels

1. Formatted messages are dispatched to various destinations to ensure visibility and record-keeping. These include email notifications for management, Slack or Teams channels for real-time SOC updates, and ticketing systems like Jira or ServiceNow for incident tracking. Additionally, these messages are often logged to the case timeline, providing a chronological, human-readable audit trail of all playbook activities.

2. Accessing Data

Data access within playbooks is the technical prerequisite for both decision-making and formatting. The platform provides a library of Python functions to manipulate and retrieve data from the SOAR environment. Operational efficiency depends on the ability to gather specific information and troubleshoot the flow of data through the playbook logs.

2.1 phantom.collect2() and phantom.debug()

1. The `phantom.collect2()` function is the primary tool for gathering specific data points from artifacts, action results, or custom fields. To facilitate the development and maintenance of these playbooks, the `phantom.debug()` function allows users to print variables and data directly into the playbook logs. This is a primary tool for troubleshooting, as it provides visibility into the data being processed at each stage of the workflow.

2.2 JSON Access and Variable Scope

1. Data in SOAR is typically structured as JSON, requiring the use of dot notation for deep inspection, such as `action_result.summary.status`. A critical distinction exists regarding variable scope: local variables in code blocks remain within those blocks. For data to persist across different blocks, developers must implement a Standard Operating Procedure (SOP) using `phantom.save_run_data()` to store variables in the global context, which can then be retrieved in subsequent Format or Code Blocks using `phantom.collect2()`.

3. Practical Use Cases for Formatted Output

1. The practical application of formatted output includes displaying summary findings to analysts within interactive prompts, which facilitates informed decision-making. Furthermore, formatting is used to log concise activity summaries to the case timeline—such as noting that a specific IP was blocked—and to generate structured incident reports for external compliance and management review.
2. The necessity of precise data access ensures that every decision made by the platform is grounded in accurate information. This foundation of data access is what enables the construction of reliable, automated playbooks.

4. Formatted Output and Data Access Practice Question

Q1: Which of the following best describes the purpose of a Format Block in Splunk SOAR?
A. It performs variable type conversions inside code blocks
B. It generates human-readable messages using templates and variables
C. It applies filters to remove sensitive data from event logs
D. It extracts threat intelligence data from artifacts

Q2: What does the `{artifact:*.cef.destinationAddress}` syntax represent in a Format Block?
A. A reference to custom Python code stored in a container
B. A variable that dynamically inserts IP address values from event artifacts
C. A static value that is hardcoded by the analyst
D. A function that replaces all values from all blocks into a single string

Q3: Which of the following is a recommended use for a Format Block's output?
A. Encrypting sensitive event fields prior to export
B. Running background jobs in the Splunk backend
C. Authenticating to external APIs via credentials
D. Logging summaries to the case timeline or sending messages to Slack

Q4: What is the benefit of using multiline formatting in a Format Block?
A. It improves execution speed for playbook completion
B. It compresses the data output from playbooks
C. It creates structured summaries or alerts with improved readability
D. It makes logs easier to encrypt before sending

Q5: Why would an analyst use the `phantom.collect2()` function in a code block?
A. To retrieve data from artifacts, action results, or custom fields
B. To schedule recurring playbook tasks
C. To register a new asset into the system dynamically
D. To create new users based on prompt responses

Q6: Which of the following best describes the use of `phantom.debug()` in a custom code block?
A. It dynamically assigns case owners
B. It outputs custom log messages to help troubleshoot playbooks
C. It replaces variables from the format block with real data
D. It sends alerts to external systems

Q7: How can JSON-like data be accessed within a Python block in Splunk SOAR?
A. Through hardcoded variables only
B. Using predefined user forms
C. Using HTML formatting functions
D. Via indexing and key-based lookup, e.g., `result[0]['summary']['status']`

Q8: What is the purpose of combining a Format Block with a user prompt?
A. To trigger a restart of the prompt after timeout
B. To let the playbook execute silently in the background
C. To enrich the playbook with new containers
D. To provide context so the user sees a readable summary before responding

Q9: What could be a reason to store formatted output in the case timeline?
A. To delay execution until the next analyst logs in
B. To automatically delete low-severity containers
C. To maintain an auditable record of key actions and findings
D. To isolate internal variables from being printed

Q10: Which of the following best demonstrates accessing the `risk_score` from a playbook action result?

A. `action_result.get("data", [{}])[0].get("risk_score")`

B. `{artifact:*.cef.risk_score}`

C. `phantom.debug("score")`

D. `run("collect_data").risk_score`

---

9. SPLK-2003 Introduction to Playbooks

Playbooks represent the "assembly line" of the SOAR factory, serving as the foundational automated workflows that ensure consistent and high-speed incident response. By codifying institutional knowledge into a predefined structure, playbooks allow security teams to respond to threats with accuracy and transparency, eliminating the variability and fatigue inherent in manual processes.

1. Core Concepts of Playbooks

A playbook is composed of four pillars: Triggers, Actions, Decisions, and User Prompts. These components interact to automate the entire lifecycle of an event, from initial ingestion to final resolution. This structure ensures that every security event is handled according to a standardized, logical sequence of steps.

1.1 Triggers and Execution Types

1. The trigger defines the starting point of a playbook. Automatic triggers are data-driven, initiating execution based on event types, tags (such as "ransomware"), or specific data properties like severity level. In contrast, manual triggers are analyst-controlled and are typically utilized when human judgment or a secondary investigation is required. While automatic triggers ensure immediate response, manual triggers provide the flexibility needed for human-in-the-loop workflows.

1.2 Actions and Asset Dependencies

1. Actions are the operational steps executed by the playbook, such as blocking an IP or querying a reputation service. These actions are powered by apps and executed via configured assets. Architects must understand that actions have a strict dependency on assets; if a required asset is misconfigured, lacks permissions, or has connection issues, the associated action block will fail at runtime, potentially interrupting the entire playbook execution.

1.3 Decisions and User Prompts

1. Decisions serve as the logic blocks of the playbook, utilizing if-then-else conditions to determine the execution path based on data results. When automation cannot make a final determination, User Prompts introduce a human-in-the-loop element. Prompts pause the playbook to gather analyst input, such as

selecting a severity level or approving a quarantine action, ensuring that high-stakes decisions remain under human control.

2. Playbook Scopes and Benefits

The platform distinguishes between Global and Local playbooks to balance reusability with specialization. The primary organizational benefits of using playbooks include consistency in process execution, increased speed to containment, and comprehensive auditability.

2.1 Global vs. Local Playbooks

1. Global playbooks are universal and reusable, designed for broad tasks like IP enrichment that apply across various event types and containers. Local playbooks are narrow and context-specific, often tied to a particular use case, container type, or specific asset configuration, such as a phishing playbook that relies on a specific email source.

2.2 Logging and Troubleshooting

1. Playbook execution activity is recorded in the primary log file located at `/var/log/phantom/playbook.log`. This log provides detailed insights into block-level performance and results, recording input/output data and tracebacks. It is the essential resource for diagnosing failures and can be combined with `phantom.debug()` output for in-depth inspection during the development phase.
2. By serving as a record of truth, the playbook ensures that every action taken is logged and time-stamped, providing the foundation for the advanced logic and filtering mechanisms required for smart automation.

3. Introduction to Playbooks Practice Question

Q1: What is the primary function of a playbook in Splunk SOAR?
A. To store logs of external security tools
B. To define user permissions and access roles
C. To automate the response to security events and cases
D. To configure dashboards and reports

Q2: Which component in a playbook determines when the playbook starts executing?
A. Action
B. Decision
C. Prompt
D. Trigger

Q3: In a Splunk SOAR playbook, what is an "Action"?
A. A logic gate that decides path routing
B. An input request sent to the user
C. An API call to an external tool using an asset
D. A built-in setting for adjusting system time

Q4: Which of the following best describes a “Decision” in a playbook?
A. A prompt asking for user input
B. A logic block that routes the workflow based on data
C. A built-in timer to pause execution
D. A rule to assign playbooks to users

Q5: What role do “User Prompts” serve within a playbook?
A. They display playbook runtime errors to the admin
B. They allow analysts to manually change system settings
C. They provide real-time status of data ingestion
D. They request input from users before continuing playbook actions

Q6: A playbook that is designed to work with any case or container type is called:
A. Modular playbook
B. Event-based playbook
C. Local playbook
D. Global playbook

Q7: What is a key characteristic of a local playbook?
A. It can only be triggered by a system reboot
B. It applies universally to all asset types
C. It is linked to specific use cases or container types
D. It cannot be edited by administrators

Q8: What is one key benefit of using playbooks in security operations?
A. They remove the need for any human analyst
B. They completely replace SIEM platforms
C. They improve consistency and auditability in incident response
D. They automatically install new apps in SOAR

Q9: Which of the following could act as a trigger for a playbook?
A. The completion of a dashboard export
B. A change in case severity
C. A system reboot
D. Asset configuration updates

Q10: Why would you use a user prompt with a dropdown menu in a playbook?
A. To collect consistent, structured input from the analyst
B. To allow the playbook to change roles dynamically
C. To bypass the need for role-based access
D. To automate case closure without confirmation

---

10. SPLK-2003 Logic, Filters, and User Interaction

The evolution of SOAR involves a shift from "fast" automation to "smart" automation through the application of sophisticated logic and data filtering. These mechanisms allow playbooks to mimic the decision-making process of a human analyst, handling complex scenarios with precision while ensuring that the platform operates efficiently even at high scales.

1. Logic and Decision Blocks

Decision Blocks are the primary logic elements used to route playbook flow based on specific conditions. Architects must master the application of comparison operators and nested logic to construct resilient logic branches. Crucially, the configuration of default paths and error branches serves as a safety net, preventing playbook stalling or crashes when encountering missing or unexpected data.

1.1 Comparison Operators and Multi-path Branching

1. Decision blocks support a range of operators, including equals (==), greater than (>), and "in" for list membership. While they often handle binary logic, they are capable of multi-path branching. This allows a single block to route an incident to different paths based on whether a severity level is High, Medium, or Low, rather than requiring multiple separate decision nodes.

1.2 Error Handling and Default Paths

1. To ensure automation resilience, a default path must be configured to prevent the playbook from stalling if no other conditions are met. Furthermore, error branches allow the playbook to handle malformed data gracefully. A playbook without an error branch is a liability in a production environment; therefore, routing errors to a specific handling path—such as an analyst alert—is considered a non-negotiable best practice.

2. Filtering Mechanisms

Filtering is essential for maintaining accuracy and performance by ensuring that playbooks only process relevant data. This reduces the number of operations performed, which is critical for scalability and managing the costs associated with external API usage and platform load.

2.1 Artifact vs. Action Filters

1. Artifact filters isolate specific data items early in the flow, such as processing only artifacts where the type is "IP Address." Action filters, however, are applied immediately before an action is executed to prevent redundant operations, such as performing a reputation lookup only if the file hash is not already known. Together, these filters ensure that the playbook focuses its resources only on necessary data points.

2.2 Advanced Filtering in Code Blocks

1. While visual filters cover basic comparisons, Python-based code blocks allow for more complex filtering. Developers use these blocks for advanced requirements like Regex parsing or processing nested dictionary structures that are not supported by the standard visual interface. This allows for a finer degree of control over what data is passed to subsequent blocks.

3. User Interaction and Prompts

Advanced user interaction features ensure that human judgment is integrated seamlessly into automated workflows. These features include highly configurable prompts that can be localized for global SOC environments.

3.1 Prompt Configuration and Localization

1. Prompts can be assigned to specific users, roles, or teams to ensure the right person makes the decision. In global or MSSP environments, the platform supports multi-language customization. This allows prompt text and dropdown choices to be localized for analysts in different regions, ensuring clarity and consistency across international teams.

3.2 Timeout Handling

1. To avoid indefinite pauses in a workflow, prompt blocks should be configured with maximum wait times and fallback actions. If a timeout occurs, the playbook can be set to escalate the case or proceed with a default decision. This maintains the momentum of the incident response and prevents a single unaddressed prompt from halting critical automation.
2. These logic and interaction features bridge the gap between automation and human judgment. However, the reliability of these features depends on the underlying stability of the system, which requires consistent maintenance.

4. Logic, Filters, and User Interaction Practice Question

Q1: Which of the following statements best describes the function of a Decision Block in a Splunk SOAR playbook?
A. It allows the playbook to evaluate conditions and take different paths based on logic
B. It configures user permissions before executing playbooks
C. It performs data extraction from incoming artifacts
D. It triggers external integrations through REST APIs

Q2: What is the purpose of using comparison operators like `==`, `in`, and `is empty` within a Decision Block?
A. To visually group playbook blocks
B. To define conditions used in branching logic
C. To select a default asset configuration
D. To assign roles to analysts

Q3: Why might you use nested Decision Blocks in a playbook?
A. To manage multi-step or dependent conditions
B. To avoid needing to use filters
C. To control access based on user role
D. To rerun a playbook after closing

Q4: Which benefit is directly associated with using artifact filters in a playbook?
A. They disable logging for irrelevant data
B. They increase the frequency of prompt usage

C. They help reduce unnecessary processing by targeting only relevant artifacts
D. They generate new event types from containers

Q5: What is a common reason to use action filters in playbooks?
A. To randomize which action is triggered
B. To ensure actions run only when certain conditions are met
C. To increase parallel processing of actions
D. To bypass asset authentication

Q6: Which of the following is an advantage of using filters in SOAR playbooks?
A. It ensures user prompts are always required
B. It increases memory consumption for complex playbooks
C. It improves performance by reducing unnecessary actions
D. It forces every artifact to be enriched

Q7: What does a prompt block do in a playbook?
A. It generates a case template for user assignment
B. It pauses the playbook to get human input
C. It sends an automated alert to the firewall
D. It merges artifacts from different cases

Q8: Why might you configure a delay in a playbook?
A. To terminate a playbook session
B. To run playbooks more than once
C. To remove artifacts from a container
D. To wait for external systems or analyst review before continuing

Q9: What is the benefit of assigning a prompt to a specific role or user group?
A. Ensures consistent scheduling of playbooks
B. Reduces performance impact by limiting visibility
C. Increases the number of filters applied
D. Directs the question to the correct personnel for decision-making

Q10: What could happen if no default path is defined for a Decision Block and no conditions are met?
A. The playbook continues as normal
B. The playbook loops back to the start
C. The playbook stalls or stops unexpectedly
D. The playbook sends a report to Splunk

---

11. SPLK-2003 System Maintenance

Maintaining a production SOAR environment requires regular system care to ensure high availability and optimal performance. Regular maintenance is not merely a "good practice" but the difference between a resilient SOC and a total outage during a high-priority incident. This involves real-time monitoring, disciplined backup routines, and proactive troubleshooting of service dependencies.

1. Monitoring and Health

The platform offers internal and external monitoring tools to track the health of critical services. Understanding the dependency chain is essential: the core `phantomd` service requires both `postgresql` and `nginx` to be fully functional. A failure in `nginx` will render the Web UI unavailable even if the underlying `phantomd` automation engine is still active.

1.1 The Health Dashboard and Internal Services

1. The System Health dashboard, located at `Administration > System Health`, provides real-time indicators for CPU, memory, disk space, and service status using a color-coded warning system. For automated monitoring, architects should poll the Health API endpoint at `https://<phantom_hostname>/rest/system_health` to ingest health metrics into external monitoring platforms.

1.2 Log Analysis and External Monitoring

1. Core logs are maintained in `/var/log/phantom/`, including `phantomd.log`, `playbook.log`, and `api.log`. Best practices involve using external tools like Splunk or Nagios to parse these logs for critical keywords like "service failed" or "disk full," providing automated alerts that bypass the platform if it becomes unresponsive.

2. Backup, Restore, and Updates

1. Protecting SOAR data involves regular backups of playbooks, app configurations, and cases. In production environments, these should be performed daily. The specific utility for this process is located at `/opt/phantom/bin/phantom_backup.sh`. It is a mandatory requirement to perform a full system backup before applying any platform updates or patches to ensure a fallback path exists should the update fail.

3. Troubleshooting and Support

A variety of diagnostic tools are available to help administrators resolve issues. These include command-line utilities for service verification and the generation of a Support Package for deep-dive technical assistance.

3.1 Command Line Utilities

1. Administrators use utilities like `phantom-syslog` for log analysis and `phenv` for executing commands within the SOAR shell environment. The command `systemctl status phantomd` is the primary method for verifying core service availability. These tools allow for diagnostics even when the Web UI is inaccessible.

3.2 Support Package Generation

1. When technical assistance is required, a Support Bundle can be generated via the UI or the CLI using the `phantom_generate_support_package.pyc` script. This bundle collects logs and configuration files into a single file, which is stored by default at `/opt/phantom/support/`. This bundle is the primary resource used by Splunk Support to diagnose complex platform issues.
2. Proactive maintenance ensures platform stability, allowing the security team to focus on building and refining automation within the Visual Playbook Editor.

4. System Maintenance Practice Question

Q1: Where can administrators monitor Splunk SOAR's CPU usage, memory load, and service status in real time?
A. On the System Health dashboard
B. Through the Asset configuration page
C. Via the Playbook Execution panel
D. Inside each case's investigation timeline

Q2: What is the purpose of the `phantomd.log` file in Splunk SOAR?
A. To track user authentication attempts
B. To record core service activity and errors
C. To monitor scheduled tasks in cron
D. To list all playbook definitions in YAML format

Q3: Which command should an administrator use to check the status of the SOAR core service?
A. `systemctl check phantom`
B. `phantom status -v`
C. `systemctl status phantomd`
D. `phantom service start`

Q4: What items should be included in a standard SOAR system backup?
A. Only playbooks and API logs
B. System logs and threat feeds
C. Just user credentials and IP whitelists
D. Playbooks, apps, assets, cases, and system settings

Q5: Before applying an update or patch to Splunk SOAR, what is the most important first step?
A. Modify all active playbooks
B. Run a test playbook on all assets
C. Create a full system backup
D. Generate a new support package

Q6: Which directory contains the primary system log files for SOAR?
A. `/var/log/phantom/`
B. `/usr/lib/logs/soar/`

C. `/opt/phantom/bin/logs/`
D. `/etc/phantom/logfiles/`

Q7: Which utility is used to safely run Splunk SOAR CLI commands with environment variables loaded?
A. `soar-runner`
B. `phantom-cli`
C. `phenv`
D. `sysrun`

Q8: What is the recommended frequency for backups in a production SOAR environment?
A. Once every 3 months
B. Daily or weekly
C. Hourly
D. Only after installing updates

Q9: What does the “Generate Support Package” tool include?
A. Only logs from `/var/log/phantom/`
B. All playbooks in JSON format
C. Logs, configuration files, and system health stats
D. Encrypted credential files only

Q10: If the SOAR web interface is unreachable but `phantomd` is running, what service should be checked next?
A. `postgresql`
B. `splunkd`
C. `nginx`
D. `phantom-syslog`

---

12. SPLK-2003 Visual Playbook Editor

The Visual Playbook Editor is a drag-and-drop interface designed to facilitate clarity, collaboration, and rapid prototyping of automation flows. By removing the need for Python expertise, it empowers the entire security team to build and manage workflows visually, making automation more accessible and transparent.

1. The Canvas and Node Types

The workspace consists of a visual canvas where nodes are arranged and connected to define the execution order. There are five primary node types—Start, Action, Decision, Prompt, and Format—which serve as the building blocks for every automated process.

1.1 The Start Block and Triggers

1. The Start Block is the mandatory entry point for every playbook. It is where the trigger type is defined—whether the playbook runs on ingestion, a manual action, or a set schedule. It also houses optional filter conditions that ensure the playbook only executes when specific criteria, such as container type or severity, are met.

1.2 Specialized Node Functions

1. The remaining nodes fulfill specialized roles: Action blocks execute tasks via assets, Decision blocks handle branching logic, Prompt blocks request human input, and Format blocks prepare human-readable messages. Collectively, these nodes allow for the creation of smart, professional-grade automation within a single visual environment.

2. Operational Lifecycle of a Playbook

1. A critical aspect of the playbook lifecycle is the "Save and Publish" process. Saving stores current edits, but the playbook is not operational until it is published. Architects must note that unpublished playbooks are invisible to automation triggers and manual execution menus; they will not run on ingested events until they are formally published.

3. Best Practices for Visual Design

1. Efficient visual design emphasizes a linear flow to avoid "spaghetti" diagrams that are difficult to audit. Administrators should use text labels and block descriptions to document the logic for other team members. Finally, every playbook must be tested with simulated data or test containers to verify behavior before it is deployed into a production environment.
2. The Visual Playbook Editor consolidates all platform features into a single interface, providing a manageable and transparent environment for the modern security automation professional.

4. Visual Playbook Editor Practice Question

Q1: What is the main purpose of the Visual Playbook Editor in Splunk SOAR?
A. To build playbooks using a visual, no-code interface
B. To execute Linux commands directly on containers
C. To configure user permissions and access controls
D. To deploy Splunk apps across multiple tenants

Q2: Which block in the Visual Playbook Editor is always the first block and defines when the playbook begins?
A. Format Block
B. Start Block
C. Action Block
D. Schedule Block

Q3: Which block in a visual playbook is responsible for executing tasks using an asset?
A. Decision Block
B. Format Block

C. Action Block
D. Prompt Block

Q4: What is the function of the Decision Block?
A. To send email alerts to analysts
B. To define the order of block execution
C. To save audit logs for reporting
D. To evaluate logical conditions and branch the workflow

Q5: Which statement accurately describes the Prompt Block in a visual playbook?
A. It runs an external Python script before continuing
B. It pauses the playbook until a user provides input
C. It logs data into a separate container
D. It auto-assigns analysts based on severity

Q6: What is the purpose of the Format Block in a Splunk SOAR visual playbook?
A. To configure asset authentication parameters
B. To transform data for human-readable output
C. To trigger REST API calls manually
D. To check for missing case tags

Q7: Which of the following is considered a best practice when using the Visual Playbook Editor?
A. Use circular node paths for fast execution
B. Group logic blocks tightly without whitespace
C. Keep flows linear and add descriptive labels
D. Use only one playbook for all use cases

Q8: What happens if no response is received for a Prompt Block within the expected time?
A. The prompt automatically selects a default value
B. The playbook continues with a “skip” branch
C. The prompt is escalated to Splunk Support
D. The playbook stays paused until input is provided

Q9: How are blocks connected in the Visual Playbook Editor?
A. Using built-in API bindings between apps
B. With drag-and-drop connections between nodes
C. Through manual Python scripting in each block
D. Using system-defined templates only

Q10: Why is it recommended to test playbooks using simulated data before going live?
A. To reduce memory usage
B. To eliminate licensing restrictions
C. To verify logic accuracy and output correctness
D. To create new asset credentials

13. SPLK-2003 Configuring External Splunk Search

The establishment of bidirectional connectivity between Splunk SOAR and Splunk Search serves as the fundamental architectural pillar of a modern Security Operations Center (SOC). By integrating these platforms, organizations transform static, isolated alerts into dynamic investigations where the automation platform can actively interrogate the data lake. This connectivity is not merely a data bridge but a strategic shift that moves the operational burden from manual lookups to automated enrichment, allowing SOAR to ask questions and Splunk to provide the context required for rapid decision-making.

This integration shifts the paradigm of security analysis by automating the heavy lifting of data retrieval. Instead of an analyst manually pivoting between consoles to correlate logs or verify historical trends, the SOAR platform programmatically retrieves necessary telemetry. This ensures that by the time a human analyst interacts with a case, the relevant historical data, logs, and alerts are already present, structured, and ready for evaluation, effectively turning raw telemetry into an actionable narrative.

1. Purpose

The primary objective of connecting SOAR to Splunk is to empower the automation engine with real-time search capabilities and access to historical logs. This integration allows playbooks to perform data enrichment—adding vital context to suspicious indicators—and validation, which helps distinguish false positives from legitimate threats. By facilitating automated data correlation across disparate sources, the integration significantly increases playbook efficiency, moving beyond simple alert handling to sophisticated threat hunting. This mechanical setup serves as the prerequisite for turning raw data into actionable intelligence.

2. Configuration Steps

The technical workflow begins with installing the Splunk App for SOAR from the App Store, which provides the necessary pre-built actions and SPL querying capabilities. Once installed, an "Asset" must be created to represent the specific Splunk search head.

2.1. **Technical Parameters:** Configuration requires the Splunk Server URL (e.g., port 8089) and appropriate authentication credentials, often utilizing an API token generated from the Splunk instance.

2.2. **Resource Management:** Defining the "Query Timeout" (e.g., 60 seconds) is critical for resource management, ensuring that SOAR does not wait indefinitely for complex searches and avoids performance degradation.

2.3. **Security Scoping:** The "Index Scope" parameter must be carefully defined to restrict the SOAR user to specific indexes (e.g., wineventlog or network_traffic), maintaining the principle of least privilege while ensuring the necessary data is accessible for automation.

3. Search Usage in Playbooks

The Search Block within a playbook acts as the engine for dynamic investigation. By inserting variables from artifacts or playbook inputs directly into SPL queries, the platform can perform granular searches based on the specific attributes of an incoming event. For example, in an enrichment scenario, a playbook might automatically search for all internal communications involving a suspicious external IP. In validation scenarios, it can check for anomalous login patterns across multiple geographies. These narrative workflows demonstrate how raw telemetry is synthesized into the "story" of an incident without manual intervention.

4. Considerations and Best Practices

Effective search integration requires strict adherence to performance and security constraints. Permissions are paramount; the Splunk account must have REST API access and specific index permissions to avoid "silent" search failures. To maintain system performance, playbooks should avoid pulling large result sets that can lead to memory exhaustion or action timeouts. Best practices include using the `| head` command in SPL to limit returns, filtering by specific host or sourcetype, and leveraging summary indexing for particularly heavy queries.

5. Saved Search Invocation and Result Handling

Advanced architectures often utilize "saved searches" maintained within Splunk rather than raw SPL embedded in playbooks to centralize search logic and improve maintainability. When handling the output of these searches, whether raw or saved, reliable data extraction is achieved through specific logic patterns using the `.get("data", [])` method. However, an educator must warn that the common path `results[0].get("data", [])[0].get("field_name")` contains an inherent risk. If the search returns no data, the `.get("data", [])` method returns an empty list, and the subsequent `[0]` index access will trigger a playbook-crashing `IndexError`. Architects must validate that the list contains elements before attempting index-based extraction.

6. Troubleshooting Scenarios

Distinguishing between permission-based failures and network-layer issues is vital for rapid resolution. In Scenario 1, a "Permission Denied" error typically indicates the asset is reachable, but the Splunk user role lacks access to a specific index or command. In Scenario 2, "Connection Refused" or timeout errors point to network-layer obstacles like firewall blocks on port 8089 or DNS misconfigurations. The "Test Connectivity" feature provides immediate diagnostic value, while inspecting the `phantomd.log` can reveal specific error signatures, guiding the transition from basic connectivity to custom extensibility through code.

7. Configuring External Splunk Search Practice Question

Q1: What is the primary purpose of configuring external Splunk search in Splunk SOAR?
A. To monitor SOAR CPU and memory usage
B. To deploy playbooks to multiple containers
C. To run SPL queries and retrieve relevant log data for security automation
D. To store backup copies of investigation cases

Q2: What must be done in Splunk SOAR before configuring a Splunk asset?
A. Run a test playbook

B. Set up a proxy service connection
C. Install the Splunk App from the App Store
D. Create a user group for index access

Q3: What happens if the Splunk asset connection fails during Test Connectivity?
A. The container is locked and flagged as failed
B. An error appears, and the connection must be manually debugged
C. A backup asset is selected automatically
D. Playbook execution continues using cached results

Q4: In configuring the Splunk asset in SOAR, what is the role of the “Query Timeout” field?
A. It sets how long Splunk retains search results
B. It defines how long SOAR waits for a search response
C. It determines Splunk's user session expiration
D. It specifies the time range of the search query

Q5: Which SPL query is most appropriate for limiting search results to 100 lines?
A. `search index=main | stats count by host`
B. `search index=main | sort limit=100`
C. `search index=main | top 100`
D. `search index=main | head 100`

Q6: Why is it recommended to include time and index filters in a Splunk SPL query inside a playbook?
A. To reduce analyst decision time
B. To improve performance and reduce result size
C. To encrypt search credentials
D. To improve branding and interface clarity

Q7: What authentication method is required when configuring the Splunk asset in SOAR?
A. Username/password or API token with search privileges
B. OAuth2 handshake with container token
C. Certificate-based exchange over FTP
D. Access via syslog port

Q8: Which of the following issues is most likely if your Splunk asset is configured with an incorrect port?
A. The case status is set to "Failed"
B. The playbook terminates and restarts
C. The test connection will fail due to timeout or refusal
D. The logs will be stored in a temporary cache

Q9: What is one reason to use a saved search instead of a raw SPL query in SOAR?
A. It improves performance and reduces query complexity
B. It prevents Splunk from generating alerts
C. It encrypts playbook traffic
D. It enables automatic Splunk license renewal

Q10: Which use case best illustrates the validation function of a Splunk search in a SOAR playbook?
A. Checking if a user has logged in from multiple geographic locations in a short period
B. Logging IP addresses into a custom list
C. Creating new tags based on alert sources
D. Sending summary reports to Slack

---

14. SPLK-2003 Custom Coding

The strategic importance of Python extensibility within Splunk SOAR cannot be overstated, as it represents the bridge between standardized app capabilities and the bespoke requirements of a complex enterprise. While visual blocks provide high-level efficiency, custom coding allows security architects to manipulate data and orchestrate workflows at a level of granularity that native integrations cannot always reach. This capability ensures that the platform remains adaptable to unique internal tools and highly specific security logic.

Custom coding fundamentally addresses the "last mile" problem in security automation. By leveraging the sandboxed Python environment, organizations can build custom data parsers, implement complex decision-making loops, and integrate with internal systems that lack official app support. This flexibility ensures that automation is never halted by the limitations of a graphical interface, allowing for a fully customized and resilient response ecosystem that scales with organizational maturity.

1. Purpose and Key Components

Custom coding in SOAR is categorized into three primary methods, each serving distinct architectural roles. Custom Functions are reusable Python components built once to perform tasks like data normalization across multiple playbooks, promoting a modular "write once, use many" design. Playbook Scripts, or Code Blocks, are embedded directly into visual workflows to provide fine-grained control for specific tasks like string manipulation or artifact looping. Finally, App Development involves building full Python integrations using YAML metadata and action handlers, which is the most robust method for connecting SOAR to internal or unsupported third-party tools.

2. Use Cases for Advanced Logic

Python is essential when visual decision blocks reach their functional limits. Advanced use cases include complex string manipulation, such as extracting usernames from unstructured logs, and the implementation of dynamic loops and fallback logic. This extensibility is particularly useful for data normalization across disparate formats like XML or YAML and for handling sophisticated API authentication methods like OAuth. Where a visual filter might fail to parse a unique data structure, a Python block can reshape that data to meet the exact needs of the downstream workflow.

3. Best Practices and Input/Output Management

Maintaining code quality and transparency is critical for long-term system health. All custom code must sanitize input data to prevent injection vulnerabilities and utilize `phantom.debug()` to provide diagnostic visibility into variable states. For Custom Functions, architects must define clear registration metadata, including the Title, Category, and Output Schema. This metadata is not just documentation; it is the structural requirement that enables the "plug-and-play" experience in the visual editor, allowing downstream blocks to reliably consume the structured dictionary returned by the function.

4. Exception Handling and Environment

Resilient automation requires robust exception handling using try/except blocks combined with `phantom.error()` to prevent unhandled exceptions from crashing a playbook run. It is also important to recognize the security boundaries of the execution environment; Playbook Code Blocks run in a sandboxed Python 3.x interpreter that restricts system-level commands like `os.system()`. In contrast, Custom Apps must explicitly declare required permissions in their YAML metadata. While code blocks are safe for logic and API requests, they operate within a restricted SOAR runtime with limited file system access.

5. Custom Coding Practice Question

Q1: What is the main purpose of using custom code in Splunk SOAR?
A. To extend automation capabilities beyond visual tools using Python
B. To set user permissions for different roles
C. To monitor memory usage on the SOAR appliance
D. To configure firewall integrations via GUI

Q2: What are Custom Functions used for in Splunk SOAR?
A. To query Splunk indexes from SOAR
B. To write reusable Python logic that can be used across playbooks
C. To generate dashboards in the Splunk UI
D. To manage asset authentication

Q3: What is a typical use case for a code block in a visual playbook?
A. Sending email notifications
B. Looping through a list of artifacts and filtering values
C. Assigning tasks to analysts
D. Logging in to the SOAR platform via REST API

Q4: What file defines actions, inputs, and metadata for a custom SOAR app?
A. `phantomd.conf`
B. `requirements.txt`
C. `metadata.yml`
D. `splunkd.log`

Q5: Which Python library is commonly used for making HTTP requests inside custom code?
A. `time`
B. `requests`

C. `os`
D. `math`

Q6: What kind of output should a Custom Function return?
A. An HTML report
B. A binary object
C. A syslog message
D. A dictionary or JSON object

Q7: Which scenario is best handled using a Custom Function instead of a visual block?
A. Checking if an IP address appears in a list
B. Converting a list of domains into top-level domains (TLDs)
C. Sending a notification email
D. Triggering a playbook on a schedule

Q8: What is the purpose of `phantom.debug()` in a code block?
A. To create new users
B. To print log messages for debugging
C. To restart playbook services
D. To simulate API responses

Q9: When building a custom app, which component handles execution of logic for a defined action?
A. Asset configuration
B. Phantom user interface
C. Action handler Python script
D. REST API CLI client

Q10: What is a recommended best practice when using user-provided input in Python code blocks?
A. Always print it to the logs
B. Convert it to a string
C. Sanitize and validate the input
D. Save it in a custom field

---

15. SPLK-2003 Custom Lists and Data Routing

Achieving context-aware decision-making requires the integration of internal datasets that reflect the organization's unique environment. Custom Lists and Data Routing provide the framework for this intelligence, allowing playbooks to differentiate between internal assets and external threats. By utilizing these internal reference tables, the SOAR platform can move beyond generic alert handling to make nuanced decisions that respect the operational context of the business.

The strategic value of these internal datasets lies in their ability to act as automation flags. By defining whitelists of approved domains or ranges of internal IP addresses, organizations can prevent automation fatigue and avoid taking destructive actions against trusted assets. This internal context is the differentiator between a noisy, generic SOC and an optimized, intelligence-driven automation program that prioritizes risks based on environmental relevance.

1. Custom Lists Management and Access

Custom Lists serve as internal reference tables within Splunk SOAR, used to store datasets like "Known Bad Domains" or "Executive User Lists." These lists are managed via the Administration UI and can be accessed in playbooks through Decision blocks or the `phantom.get_list()` function. Architects must be mindful of structural constraints; lists are flat and do not support nested JSON. Performance may degrade if a list exceeds 10,000 rows. Furthermore, SOAR treats empty cells as empty strings; therefore, developers must implement validation in code blocks to skip these empty rows during processing.

2. Data Routing Rules and Dynamic Logic

Data routing is the mechanism that directs events to the appropriate response path based on content and context. This is achieved through ingestion settings, case templates, and the strategic use of tags and labels. Labels categorize containers at a high level (e.g., "endpoint alert") to trigger the correct investigation playbook. Tags act as free-form automation flags that can trigger specific behavior, such as auto-closure logic for known false positives or triggering immediate escalation workflows for "VIP" users.

3. Advanced Limitations and Dynamic Asset Selection

In advanced workflows, dynamic asset selection can be implemented using decision blocks and `phantom.save_run_data()`. This allows a playbook to evaluate event context—such as identifying whether an IP is internal or external—and dynamically choose the correct security asset (e.g., a specific firewall instance) for a response action. This methodology ensures that the automation remains flexible, though architects must ensure the dynamically selected asset names exactly match the defined asset names in the platform to avoid action failures.

4. Custom Lists and Data Routing Practice Question

Q1: What is the primary use of a custom list in Splunk SOAR?
A. To store real-time log data from containers
B. To define alert thresholds for CPU and memory
C. To store reference values for use in playbook logic
D. To monitor system health and service status

Q2: Where can a Splunk SOAR administrator create and manage custom lists?
A. Administration > Custom Lists
B. Case Overview > Settings
C. Playbook Editor > Format Block
D. Dashboard > Asset Panel

Q3: Which of the following is a valid use case for a custom list?
A. Running an automatic backup of playbooks
B. Querying Splunk search logs for every event
C. Whitelisting known good sender domains
D. Creating a container from a raw alert

Q4: What is one of the key benefits of dynamic data routing in Splunk SOAR?
A. It guarantees the use of the same asset for all alerts
B. It allows routing decisions based on incoming data content
C. It removes the need for playbooks altogether
D. It prevents the execution of decision logic in child playbooks

Q5: Which scenario best illustrates how a custom list improves playbook accuracy?
A. Comparing IPs to a predefined list of internal subnets
B. Assigning all events to the same team
C. Changing the event label to “urgent”
D. Sending all notifications to a manager’s email

Q6: What is the recommended way to check whether a value exists in a custom list using Python?
A. phantom.log("check")
B. phantom.get_list("ListName")
C. phantom.set_param("value")
D. phantom.filter("value in list")

Q7: How are routing rules typically defined in Splunk SOAR?
A. Through Format Blocks in case summaries
B. Within ingestion settings and case templates
C. Inside log rotation policies
D. By manually editing container UUIDs

Q8: Which of the following best describes the role of labels in data routing?
A. They identify the app that should process an event
B. They store dynamic variables for format blocks
C. They trigger health alerts based on list values
D. They classify containers to control routing and playbook selection

Q9: When should you use tags instead of custom lists?
A. When the data source requires API key-based access
B. When you want to persist system logs
C. When you want flexible, analyst-added classification for routing
D. When working with nested child playbooks

Q10: Which of the following is a feature of multi-column custom lists in Splunk SOAR?
A. They require CSV output formatting in playbooks
B. They only work with the REST API
C. They allow values to have multiple attributes like name and reason
D. They cannot be accessed from decision blocks

---

16. SPLK-2003 Integrating SOAR into Splunk

The "closed-loop" security workflow is realized through the bidirectional flow of data between Splunk, acting as the detection engine, and SOAR, acting as the response engine. This integration ensures that the security lifecycle is continuous, where a detection in Splunk triggers an immediate response in SOAR, which then feeds results back into Splunk for documentation and dashboarding. This synergy minimizes the mean time to respond by automating the handoff between discovery and remediation.

This bidirectional integration optimizes the detection-to-remediation timeline by removing manual pivot points. When Splunk pushes a notable event into SOAR, the context of the search is preserved, allowing the automation engine to start enrichment and investigation without human delay. The subsequent feedback loop ensures that the SOC's primary visibility tool—Splunk—is always updated with the findings and actions taken by the automation platform, providing a unified view of the incident lifecycle.

1. Integration Points: App for SOAR and ARF

Integration is facilitated through two primary models with different interaction triggers. The Splunk App for SOAR focuses on automated ingestion, using custom alert actions to push correlation search results directly into SOAR containers. In contrast, the Adaptive Response Framework (ARF) enables a manual, contextual model where analysts can right-click a notable event in the Splunk ES interface to trigger a specific SOAR playbook for immediate investigation or remediation, such as host isolation.

2. Data Flow and Artifact Mapping

When data moves from Splunk to SOAR, each search result field is mapped into a CEF-formatted artifact within a SOAR container. For example, `source_ip` in Splunk is mapped to `cef.sourceAddress`. Adhering to the Splunk Common Information Model (CIM) during search creation is essential for cross-platform compatibility, as it ensures that standard SOAR playbooks can reliably parse and act upon the incoming telemetry without requiring custom mapping for every alert.

3. Authentication and Advanced Feedback Mechanisms

Security for this integration is maintained through API-based authentication, with a strict preference for restricted API tokens and HTTPS encryption. The feedback loop completes the lifecycle, allowing SOAR to programmatically update Splunk notable events, adjust risk scores, or write summaries back to a Splunk index. This ensures that the results of automation are visible in Splunk dashboards, providing full traceability and a single pane of glass for both detection and response history.

4. Troubleshooting Integration Failures

Resolving integration failures requires a systematic review of the ingestion path. Common issues stem from misconfigured API addresses, incorrect event label mapping, or playbook trigger modes not being set to "on ingestion." If events arrive with missing data, architects should validate that the Splunk alert action is correctly mapping all required fields to the CEF structure and that the payload is valid JSON. This diagnostic process ensures the reliability of the bridge between detection and response.

5. Integrating SOAR into Splunk Practice Question

Q1: What is the main function of the Splunk App for SOAR?
A. To send alerts to SOAR and show SOAR case data in Splunk
B. To upgrade SOAR automatically from within Splunk
C. To allow SOAR to directly modify Splunk correlation searches
D. To install SOAR on a Splunk indexer

Q2: What role does the Adaptive Response Framework (ARF) play in Splunk Enterprise Security?
A. It creates dashboards in SOAR automatically
B. It allows users to trigger SOAR playbooks from within notable events
C. It disables SOAR integrations temporarily during upgrades
D. It blocks IPs directly from the Splunk search head

Q3: Which action initiates the data flow from Splunk to SOAR during integration?
A. Manual lookup creation
B. Installing Splunk’s Machine Learning Toolkit
C. Firing a correlation search with a custom alert action
D. Writing a custom add-on

Q4: How does SOAR receive incoming events from Splunk?
A. Via forwarded syslog messages
B. Through email ingestion only
C. Via Splunk data models
D. Through the REST API and configured assets

Q5: What can a SOAR playbook do once triggered by an event from Splunk?
A. Only log the event without taking action
B. Perform enrichment, take automated actions, and send feedback
C. Restart the Splunk instance for further analysis
D. Edit Splunk dashboards directly

Q6: What is one benefit of viewing SOAR case status from within Splunk?
A. It allows deletion of SOAR playbooks from Splunk
B. It improves log ingestion rates
C. It provides end-to-end visibility without switching platforms
D. It reduces Splunk storage usage

Q7: Why might an alert fail to reach SOAR from Splunk?
A. Incorrect dashboard permissions
B. Disabled correlation search tagging

C. SOAR asset has too many users
D. Misconfigured alert action or API authentication

Q8: How can SOAR enhance Splunk's detection capabilities?
A. By adjusting data retention policies in Splunk
B. By creating additional Splunk correlation searches
C. By enriching and escalating events using external intelligence
D. By modifying the Splunk index structure

Q9: What is the best way to reduce latency in alert forwarding from Splunk to SOAR?
A. Use scheduled searches once a day
B. Use real-time correlation searches with custom alert actions
C. Rely on manual JSON exports
D. Disable the Splunk App for SOAR

Q10: What key information must a custom alert action include when sending to SOAR?
A. Relevant fields like IPs, usernames, artifacts
B. Splunk license key
C. Only IP addresses
D. Case resolution time

---

17. SPLK-2003 Modular Playbook Development

Applying software engineering principles—specifically modularity—to security automation is essential for managing organizational scalability. By breaking complex workflows into smaller, task-specific units, SOCs can reduce technical debt and simplify the development process. The parent-child model allows for the creation of a sophisticated response ecosystem that is easier to maintain and faster to deploy than monolithic, single-workflow playbooks, mimicking modern application development.

A modular approach enhances organizational scalability by allowing specialized teams to manage specific child playbooks while a central orchestration parent manages the overall logic. This structure ensures that as new threats or tools emerge, only the relevant modules need to be updated. This practice ensures that improvements to a single module propagate throughout the entire ecosystem, maintaining consistency and reducing the risk of operational errors during updates.

1. Concept of the Parent-Child Structure

In a modular architecture, the Parent Playbook acts as the high-level orchestrator, controlling the overall logic and decision-making flow. it utilizes the Playbook Block to call specialized Child Playbooks, which are self-contained units designed for a single task, such as "IP Enrichment" or "Host Isolation." For the child playbook to receive

data, architects must explicitly define input parameters (e.g., target_ip) in the child's settings, which are then mapped in the parent's Playbook Block.

2. Strategic Benefits: Reusability and Maintainability

The primary benefit of modularity is the significant reduction in technical debt through reusability. If a response procedure changes, such as the method for disabling a user account, the architect only needs to update one child playbook. This change automatically propagates across every parent playbook using that module. This "write once, use many" philosophy ensures consistent response procedures and simplifies the testing of individual units of logic.

3. Best Practices for Modular Design

Effective modular design requires child playbooks to be strictly focused on single tasks and to return structured JSON data to the parent. Parent playbooks must be designed for resilience; they should use decision blocks to check the return status of a child playbook and handle failures, such as API timeouts, gracefully. Automation must be built to be resilient rather than fragile, ensuring that one failed sub-task does not halt the entire investigative workflow.

4. Governance: Sharing and Version Control

Governance in multi-tenant environments is managed through global publishing and role-based access control (RBAC). Utility playbooks should be marked as "Global" to make them available across all teams. Because SOAR does not provide automatic versioning, architects must use manual strategies: cloning playbooks for iteration, using versioned naming (e.g., `_v1`, `_v2`), and documenting all changes in the description field to ensure safe and transparent updates.

5. Modular Playbook Development Practice Question

Q1: What is the main purpose of using a Playbook Block in a parent playbook?
A. To assign case ownership based on logic
B. To call and execute a child playbook within a larger workflow
C. To display formatted messages to end users
D. To test container inputs using JSON validation

Q2: Which of the following best describes a child playbook in a modular playbook design?
A. A playbook that handles all automation logic for one use case
B. A backup playbook used when the main one fails
C. A small, reusable playbook that performs a specific task
D. A troubleshooting template used for testing actions

Q3: What should the parent playbook do when a child playbook fails due to invalid input or timeout?
A. Automatically rerun the child playbook three times
B. Stop execution and raise a critical alert
C. Ignore the result and continue the workflow
D. Use decision logic to handle the failure gracefully

Q4: What is one of the primary advantages of using modular playbooks?
A. They allow deeper access to system logs
B. They reduce the need for asset configuration
C. They improve scalability and manageability of automation workflows
D. They automatically trigger from container ingestion

Q5: Why is modular playbook development considered a best practice?
A. It ensures all playbooks are executed in under 1 minute
B. It increases reuse, simplifies maintenance, and enhances clarity
C. It eliminates the need for user prompts in automation
D. It bypasses the requirement for child playbook documentation

Q6: Which of the following is a recommended design guideline when creating a child playbook?
A. Focus the playbook on one clear task
B. Handle multiple types of artifacts and tasks
C. Include complex decision trees with fallback logic
D. Combine enrichment, containment, and reporting in one playbook

Q7: What happens when a Playbook Block calls a child playbook that returns no data?
A. The parent playbook is automatically paused
B. The asset configuration is reset
C. An email is sent to the system administrator
D. The parent playbook must handle it using conditional logic

Q8: Which of the following is a benefit of returning structured JSON data from child playbooks?
A. It improves memory usage and CPU performance
B. It allows the child playbook to auto-diagnose issues
C. It simplifies downstream data processing in the parent playbook
D. It enables more granular access control

Q9: In Splunk SOAR, when should you consider using a modular playbook approach?
A. Only for non-production workflows
B. When the logic is long, reusable, or team-shared
C. Only when using community-developed playbooks
D. When the parent playbook contains fewer than 5 blocks

Q10: What is a best practice when passing data between a parent and child playbook?
A. Only pass raw action results for transparency
B. Use clearly named inputs and structure output in JSON format
C. Avoid passing data to reduce complexity
D. Only pass internal variables from prompts

---

18. SPLK-2003 Using REST

The REST API serves as the "remote control" for the Splunk SOAR platform, providing the essential interface for programmatic interaction. This capability is strategically vital for integrating SOAR into a broader ecosystem that includes ticketing systems and threat intelligence platforms. By exposing its core functions via REST, SOAR allows developers to automate platform management and event ingestion from any system capable of making HTTPS-encrypted requests, facilitating a "SOAR as code" approach.

The programmatic flexibility offered by the REST API allows for the automation of SOAR itself. Whether it is synchronizing cases between different instances or auto-creating events from a Jira ticket, the API removes the need for manual UI interactions. This enables a level of operational scale where the platform can be managed via scripts, fitting into modern DevOps and SecOps pipelines and allowing the organization to transcend the limitations of the web interface.

1. API Capabilities and Common Endpoints

The REST API supports a range of operations via endpoints like `/rest/container` for event creation, `/rest/artifact` for retrieving indicator data, and `/rest/playbook_run` for triggering automation. These capabilities allow external systems to feed data into SOAR and monitor the results of investigations. This programmatic access is fundamental for building custom dashboards or synchronizing data across multi-tenant SOAR environments.

2. Authentication Methods and Security Practices

Access is secured through token-based authentication, which is the preferred method for scripts. Security dictates the use of HTTPS and the principle of least privilege; tokens should be scoped to the minimum roles required. Developers must implement logic to distinguish between a 401 Unauthorized error (invalid/expired token) and a 403 Forbidden error (valid token but insufficient permissions). Regular token rotation and monitoring of API logs are essential to prevent unauthorized access.

3. Handling Pagination and Rate Limiting

To protect system performance, the SOAR API utilizes pagination parameters such as `page` and `page_size`. Architects must design integration scripts to loop through these pages until the entire dataset is retrieved. While SOAR does not impose strict public rate limits, best practices involve implementing retry logic with backoff delays and monitoring for 429 status codes to avoid overwhelming the platform during high-volume operations.

4. Asynchronous Operations and Monitoring

The `/rest/playbook_run` endpoint is fundamentally asynchronous; it returns a `run_id` immediately, but the playbook execution continues in the background. A robust integration must include polling logic that uses this `run_id` to query the status—checking if the run is "in_progress", "completed", or "failed"—before attempting to capture any summary data or results generated during the run.

5. Developer Best Practices for Integration

Robust integrations require secure developer practices, such as storing API tokens in environment variables or secret management tools rather than hard-coding them. Python developers should utilize structured error handling with try/except blocks to manage connection timeouts and handle status codes gracefully. These RESTful capabilities unify the platform's diverse components—from search and custom coding to lists and playbooks—into a single, cohesive, and programmatically accessible automation ecosystem.

6. Using REST Practice Question

Q1: What is the primary purpose of Splunk SOAR's REST API?
A. To allow external systems to programmatically interact with SOAR
B. To configure Splunk indexers from SOAR
C. To build custom dashboards inside Splunk
D. To encrypt data stored in artifacts

Q2: Which authentication method is recommended for secure REST API usage?
A. Using browser cookies
B. Allowing anonymous access
C. Using admin login with plain passwords
D. Token-based authentication

Q3: What does the endpoint `POST /rest/container` do?
A. Fetches a list of users from SOAR
B. Uploads an attachment to a case
C. Triggers a playbook execution
D. Creates a new event (container) in SOAR

Q4: Which Python function is most commonly used to send REST requests when integrating SOAR with custom tools?
A. requests.get/post() from the `requests` library
B. os.system()
C. socket.request()
D. json.dumps()

Q5: What is a potential risk of not limiting API token permissions?
A. It may expose sensitive actions to unauthorized users
B. It increases memory consumption
C. It can prevent log collection from the server
D. It causes playbooks to run more slowly

Q6: What is the main function of the `/rest/playbook_run` endpoint?
A. View SOAR audit logs
B. Modify user roles and permissions
C. Upload a file attachment to an artifact
D. Trigger a specific playbook on a container or case

Q7: Why should API requests always be made over HTTPS?
A. To ensure encryption and secure data transfer
B. To allow token reuse
C. To avoid overloading the SOAR CPU
D. To enable OAuth-based login

Q8: In which situation would you use the `phantom.debug()` function?
A. To generate tokens
B. To print diagnostic information during playbook execution
C. To upload files via API
D. To reassign ownership of a case

Q9: What is a common real-world use case for using the REST API?
A. Automatically generating tickets in Jira from a SOAR event
B. Scheduling Splunk maintenance windows
C. Uploading datasets into Splunk from Excel
D. Enabling debug mode on the SOAR interface

Q10: Which tool is commonly used for interactively exploring SOAR API endpoints?
A. Grafana
B. Postman
C. Tableau
D. Nessus

Learning Path & Study Advice

A strong preparation path begins with understanding the overall Splunk SOAR platform before focusing narrowly on automation development. Candidates should first become comfortable with deployment concepts, user administration, apps, assets, investigations, and case workflows so that playbook development is understood in its operational context rather than as an isolated technical task.

From there, the next step is to build a solid conceptual foundation in playbooks: what they do, how they are structured, how data moves through them, and how workflow logic is designed. After that, study should progress into the practical mechanics of the Visual Playbook Editor, filters, branching, user interaction, and formatted output. This sequence helps develop clarity around how automation decisions are made and how results are communicated.

Once the fundamentals are clear, candidates should focus on deeper development topics such as modular design, custom lists, data routing, external Splunk search configuration, and integration patterns between SOAR and Splunk. The final stage of preparation should emphasize extension and flexibility through custom coding and REST-based interaction. Throughout the learning process, the most effective approach is to connect each topic to real operational scenarios: how analysts investigate work, how cases are managed, how data is enriched, and how automation supports response decisions. The goal should be practical comprehension, clean logic design, and confidence in how the platform behaves as a complete working system.

Who This PDF Is For

This document is intended for learners preparing for the SPLK-2003 certification who want a structured understanding of the knowledge areas covered by the exam blueprint. It is especially suitable for SOAR developers, security automation practitioners, security engineers, and technical analysts who work with orchestration workflows and platform integrations. It is also useful for administrators or implementation specialists who need to understand how platform configuration, case handling, and playbook development fit together.

The most suitable readers are those with some prior exposure to security operations processes and a basic comfort level with technical logic, workflow thinking, or scripting concepts. This PDF will be most beneficial for candidates who want an organized, professional overview of the certification scope and who prefer to study from a concept-driven perspective rather than from isolated feature memorization.

Call To Action

This document provides an overview of structured learning and certification preparation approaches. For learners seeking clear knowledge organization, guided study planning, and exam-focused practice resources, AAAdemy offers a comprehensive platform to support independent and effective learning.

Explore additional training materials, study guidance, and practice resources at:

Splunk SPLK-2003 SOAR Certified Automation Developer Certification Training Course - AAAdemy

Online Flashcards (Quizlet):

https://quizlet.com/user/AAAdemy/folders/splk-2003-splunk-soar-certified-automation-developer-exam?i=6zfa5t&x=1xqt

Attachment : Answers by Knowledge Point

Deployment, Installation, and Initial Configuration Practice Question
A1:
Answer: D
Explanation: Standalone deployment is the most suitable for small teams or labs. It is simple, requires less hardware, and is ideal for testing environments.

A2:
Answer: A
Explanation: Standalone deployments don’t scale well and if the single server fails, the entire system goes down. This makes it less ideal for larger or production environments.

A3:
Answer: B
Explanation: Cloud deployment provides flexibility, scalability, and removes the need for physical hardware. It’s ideal for companies that are cloud-native or require high availability.

A4:
Answer: D
Explanation: OVA images are pre-configured virtual machines that allow quick deployment in environments like VMware or VirtualBox. They're ideal for demonstrations and labs.

A5:
Answer: C
Explanation: Setting the correct system time and timezone is crucial, as incorrect timestamps will affect logs, investigation timelines, and correlation.

A6:
Answer: B
Explanation: OVA images are less flexible in customizing system resources and configurations, making them unsuitable for production-grade deployments.

A7:
Answer: D
Explanation: The web-based setup wizard is used to configure the admin account, SMTP server, proxy settings, licensing, and other core initial settings.

A8:
Answer: A
Explanation: Manual Linux installation gives administrators the most control, making it suitable for production environments where customization is needed.

A9:
Answer: C
Explanation: `nmtui` is a text-based user interface for NetworkManager that allows setting IP addresses, hostnames, DNS, etc., during initial configuration.

A10:
Answer: C
Explanation: SOAR allows the use of trusted CA certificates to secure web access with HTTPS, improving security and meeting compliance standards.

User Management Practice Question
A1:
Answer: D
Explanation: RBAC allows permissions to be managed through roles, which are then assigned to users. This simplifies user access control and ensures consistency.

A2:
Answer: B
Explanation: Administrators have full access and are responsible for system-level tasks such as app installation, system configuration, and user management.

A3:
Answer: B
Explanation: LDAP/AD integration allows centralized user account management, syncing credentials and groups from your organization’s directory server.

A4:
Answer: C
Explanation: Custom roles allow you to define specific permissions such as allowing playbook execution while restricting asset modification.

A5:
Answer: A
Explanation: Local authentication is best suited for small-scale deployments, as it requires manual user and password management.

A6:
Answer: B
Explanation: SAML is powerful but requires correct setup on both SOAR and the IdP side. Misconfigurations can lock users out of the platform.

A7:
Answer: C
Explanation: The Analyst role allows event interaction, playbook execution, case ownership, note-taking, and escalation — all part of active incident handling.

A8:
Answer: B
Explanation: 2FA significantly increases login security by requiring users to provide a second verification code in addition to their password.

A9:
Answer: D
Explanation: Splunk SOAR uses Role-Based Access Control (RBAC), where roles define what actions are allowed, and users inherit those by role assignment.

A10:
Answer: D
Explanation: Local authentication requires no external system configuration and works immediately after installation, making it the easiest to set up.

Apps, Assets, and Playbooks Practice Question
A1:
Answer: B
Explanation: Apps act as connectors or adapters between SOAR and external systems like firewalls, cloud providers, or threat intel tools, and they define the actions that can be called.

A2:
Answer: A
Explanation: Assets are the configured instances of apps, containing connection details such as API keys, IP addresses, and other required settings for real use.

A3:
Answer: C
Explanation: Playbooks can be triggered automatically (on event ingestion), manually (by analyst), or on a schedule.

A4:
Answer: D
Explanation: If the required asset is missing or misconfigured, the action step will fail, potentially breaking the entire playbook workflow.

A5:
Answer: B
Explanation: Playbooks can be executed automatically (via event ingestion), manually (by user interaction), or on a schedule.

A6:
Answer: D
Explanation: Assets store authentication details such as API keys, credentials, IPs, and network settings necessary for SOAR to interact with external systems.

A7:
Answer: D

Explanation: Sub-playbooks or modular playbooks enable the reuse of smaller workflows, improving maintainability and efficiency in complex automation.

A8:
Answer: A
Explanation: Format blocks help clean or structure data, making it suitable for prompts, logging, or report generation.

A9:
Answer: A
Explanation: You can configure multiple assets from the same app, such as `Splunk-Test` for dev and `Splunk-Prod` for production environments, each with different credentials/settings.

A10:
Answer: B
Explanation: “Test Connectivity” is used to ensure the asset is correctly configured and SOAR can reach and authenticate with the external service.

Analyst Queue Practice Question
A1:
Answer: A
Explanation: The Analyst Queue serves as a central hub for managing and triaging incoming security events, providing visibility and control for SOC analysts.

A2:
Answer: D
Explanation: Manual assignment helps ensure accountability and avoids situations where multiple analysts unknowingly work on the same event.

A3:
Answer: B
Explanation: Auto Assignment Rules automatically distribute incoming events based on conditions such as severity, time of arrival, or event type.

A4:
Answer: C
Explanation: Automatic assignment is managed under Ingestion Settings > Ingest Settings Rules, where conditions and destinations are configured.

A5:
Answer: B
Explanation: The given filter precisely targets phishing events that are unresolved, of high severity, and recent—ideal for senior triage.

A6:
Answer: C

Explanation: Unassigned events remain in the queue and can be claimed manually or automatically assigned by rules.

A7:
Answer: D
Explanation: Filters help analysts zero in on events that match their expertise, assignments, or investigative priorities.

A8:
Answer: A
Explanation: Events can be sorted or filtered by severity, source, tags, and time—but "Response Time" is not a standard filter.

A9:
Answer: C
Explanation: Auto Assignment ensures alerts are quickly routed to the right person or team, improving SOC response times and load balancing.

A10:
Answer: A
Explanation: Combining manual and automatic assignment ensures even workload distribution and operational efficiency in SOC environments.

The Investigation Page Practice Question
A1:
Answer: A
Explanation: The Investigation Page is where analysts interact with an individual incident, view timelines, run actions, analyze artifacts, and document findings.

A2:
Answer: C
Explanation: The timeline provides a complete history of all activity within the case, including automated and manual actions, notes, and status transitions.

A3:
Answer: B
Explanation: The Artifacts Panel shows extracted data points related to the incident and allows analysts to sort, filter, and take direct action on individual artifacts.

A4:
Answer: D
Explanation: The Manual Actions Panel allows analysts to run playbooks or specific actions on demand, often with input parameters adjusted during investigation.

A5:
Answer: C
Explanation: The Notes Section is used for documenting analyst observations, manual decisions, and shared updates throughout the incident lifecycle.

A6:
Answer: B
Explanation: Analysts often use the Artifacts Panel to trigger enrichment actions such as geolocation or threat intel lookups for IPs or file hashes.

A7:
Answer: C
Explanation: The Investigation Page allows analysts to monitor live playbook execution and manually intervene, blending automation with human decision-making.

A8:
Answer: A
Explanation: The timeline shows case-related actions. System-level changes like unrelated asset reconfigurations do not appear here.

A9:
Answer: D
Explanation: Analysts can take direct action on individual artifacts such as running targeted playbooks or checking reputation.

A10:
Answer: B
Explanation: Playbooks can pause for user input, and manual intervention ensures better decision-making when automation is insufficient.

Case Management and Workbooks Practice Question
A1:
Answer: A
Explanation: A Case in Splunk SOAR organizes all information related to a security incident and supports tracking, analysis, collaboration, and resolution workflows.

A2:
Answer: B
Explanation: Cases can be created either automatically based on event conditions (like severity or tags), or manually when an analyst determines it's needed.

A3:
Answer: D
Explanation: "Alert Volume" is not a standard case field. Valid case fields include Owner, Tags, Severity, Title, Status, and Associated Events.

A4:
Answer: C
Explanation: Case linking lets analysts consolidate related events—such as multiple endpoints affected by the same attack—into a single, unified case.

A5:
Answer: C

Explanation: Case Templates ensure consistency by predefining fields, default values like severity or tags, and associating appropriate workbook templates.

A6:
Answer: B
Explanation: Stages represent investigation phases such as Detection or Containment, and each stage is composed of one or more tasks.

A7:
Answer: A
Explanation: Manual Tasks require human action, such as writing a note or verifying data. Automated Tasks are linked to playbooks that SOAR can execute automatically.

A8:
Answer: B
Explanation: Each task and stage in a workbook can be marked as Pending, In Progress, or Completed, helping teams and managers track progress and identify bottlenecks.

A9:
Answer: C
Explanation: Organizations can create their own workbook templates, define custom stages and tasks, specify automation, and align workflows with internal policies.

A10:
Answer: B
Explanation: Tasks are specific investigative actions within workbook stages—"Query user login history" is typical of a triage task.

Customizations Practice Question
A1:
Answer: A
Explanation: Custom fields allow organizations to store and track context-specific data—such as business unit, region, or incident owner—on events, artifacts, or cases.

A2:
Answer: B
Explanation: Custom layouts let you change how fields appear in the user interface—grouping them, hiding/showing them, and tailoring the visual flow of data entry or viewing.

A3:
Answer: C
Explanation: Dynamic menus allow analysts to select from real-time data sources such as assets, user groups, or threat levels, making prompts more context-aware and accurate.

A4:
Answer: D
Explanation: Custom actions are Python-based logic blocks developers can write to extend SOAR’s functionality, enabling API calls, internal data lookups, and custom integrations.

A5:
Answer: C
Explanation: Tags are metadata labels applied to events that help with classification, filtering, and can also be used to automatically trigger playbooks or workflows.

A6:
Answer: A
Explanation: Theme and branding customization allows organizations to personalize SOAR with their logos, color schemes, and messages—especially useful for MSSPs.

A7:
Answer: B
Explanation: Webhooks provide a fast and lightweight integration point for external systems to send data directly into SOAR, triggering ingestion or workflows.

A8:
Answer: D
Explanation: Splunk SOAR allows developers to extend its REST API by creating new endpoints or modifying existing ones for internal dashboards or system integrations.

A9:
Answer: C
Explanation: Tags can be applied manually by users or automatically by playbooks and ingestion rules to categorize and act upon events.

A10:
Answer: D
Explanation: Dynamic menus can populate options in real time (e.g., from an internal tool list), giving analysts current, relevant choices during playbook execution.

System Maintenance Practice Question
A1:
Answer: A
Explanation: The System Health dashboard, accessible via Administration > System Health, provides real-time system performance indicators such as CPU, memory, disk, and service status.

A2:
Answer: B
Explanation: The `phantomd.log` file is the primary log for the SOAR platform's core service and helps administrators understand system operations and diagnose problems.

A3:
Answer: C
Explanation: To verify if the core `phantomd` service is active, the correct command is `systemctl status phantomd` or `sudo service phantomd status`.

A4:
Answer: D
Explanation: A proper backup includes all essential operational components: playbooks, assets, apps, cases, and configuration settings such as roles, custom fields, and dashboards.

A5:
Answer: C
Explanation: Backing up your system before applying updates is a best practice to prevent data loss or misconfiguration due to unforeseen issues during the upgrade.

A6:
Answer: A
Explanation: All major SOAR logs, including `phantomd.log`, `playbook.log`, and `api.log`, are located in `/var/log/phantom/`.

A7:
Answer: C
Explanation: `phenv` loads the Splunk SOAR environment and allows Python and shell scripts to interact with the platform safely and correctly.

A8:
Answer: B
Explanation: For production systems, backups should be automated to run daily or weekly to ensure data protection in the event of system failure.

A9:
Answer: C
Explanation: The Support Package feature collects logs, configuration data, and health information to assist Splunk Support in diagnosing issues effectively.

A10:
Answer: C
Explanation: `nginx` is the web server responsible for serving the SOAR web interface. If `phantomd` is running but the interface is unreachable, `nginx` could be the root cause.

Introduction to Playbooks Practice Question
A1:
Answer: C
Explanation: Playbooks are automated workflows designed to respond to security events and cases. They handle tasks such as enrichment, containment, and escalation using predefined steps.

A2:
Answer: D
Explanation: The trigger defines the condition under which the playbook starts. It could be event ingestion, manual execution, or changes in case attributes.

A3:
Answer: C
Explanation: Actions are operations that interact with external systems (e.g., block IP, query reputation) and are executed through assets configured in SOAR.

A4:
Answer: B
Explanation: Decision blocks analyze results from actions or inputs and determine which logical path the playbook should follow next, similar to "if-then-else" logic.

A5:
Answer: D
Explanation: Prompts are used when human approval or input is needed. The playbook pauses and waits for a response before proceeding.

A6:
Answer: D
Explanation: Global playbooks are reusable and can be attached to multiple events, cases, or types of containers regardless of specific context.

A7:
Answer: C
Explanation: Local playbooks are tailored to specific assets, event types, or use cases, and are typically not reused across different scenarios.

A8:
Answer: C
Explanation: Playbooks ensure each event is handled consistently and provide an auditable log of actions, enhancing both compliance and quality of response.

A9:
Answer: B
Explanation: Triggers can include changes in a case’s status, such as severity updates, or new event ingestions. These initiate the playbook automatically.

A10:
Answer: A
Explanation: Dropdowns in user prompts help standardize user input, making decisions more consistent and suitable for automated logic that follows.

Visual Playbook Editor Practice Question
A1:
Answer: A
Explanation: The Visual Playbook Editor provides a drag-and-drop, no-code interface where users can visually build and connect playbook components like actions, decisions, and prompts.

A2:
Answer: B

Explanation: The Start Block is the mandatory entry point of every playbook and defines the trigger condition (manual, ingestion, scheduled) for execution.

A3:
Answer: C
Explanation: Action Blocks are used to interact with external tools through configured assets, such as querying threat intel or blocking an IP.

A4:
Answer: D
Explanation: Decision Blocks evaluate the output of previous actions or playbook data using conditional logic (e.g., greater than, equals) to determine the next path of execution.

A5:
Answer: B
Explanation: Prompt Blocks are used to gather input from users, often through dropdowns or text fields. The playbook execution pauses until the input is received.

A6:
Answer: B
Explanation: The Format Block allows users to structure and format data into human-readable strings, which can then be sent via email, logged, or added to notes.

A7:
Answer: C
Explanation: Clear, linear flows and meaningful labels help make the playbook readable and maintainable, especially for collaborative or long-term usage.

A8:
Answer: D
Explanation: A playbook halts execution at the Prompt Block and will not resume until a user provides the required input, unless timeout logic is explicitly built in.

A9:
Answer: B
Explanation: Blocks are connected visually by dragging arrows from one block to another, establishing the order and logic flow.

A10:
Answer: C
Explanation: Testing with simulated or test containers allows users to confirm that logic, formatting, and action responses behave as expected before live deployment.

Logic, Filters, and User Interaction Practice Question
A1:
Answer: A
Explanation: A Decision Block is used to evaluate logical conditions (e.g., IP reputation score, case severity) and direct the flow of the playbook based on those evaluations.

A2:
Answer: B
Explanation: Comparison operators are used inside Decision Blocks to build condition logic. For example, `==` checks for equality, while `in` checks membership in a list.

A3:
Answer: A
Explanation: Nesting Decision Blocks allows you to evaluate additional conditions only if a prior condition is met, enabling complex, layered automation logic.

A4:
Answer: C
Explanation: Artifact filters allow the playbook to focus only on artifacts that meet specific criteria, such as type or risk score, which improves accuracy and reduces noise.

A5:
Answer: B
Explanation: Action filters evaluate criteria before an action is executed (e.g., only block IPs with risk score > 70), ensuring the action is meaningful and efficient.

A6:
Answer: C
Explanation: Filters improve playbook performance by limiting the scope of data and actions, avoiding redundant or irrelevant automation steps.

A7:
Answer: B
Explanation: Prompt blocks interrupt playbook execution and wait for user input, such as selecting a response, entering notes, or choosing from dropdown options.

A8:
Answer: D
Explanation: Delay blocks are used to pause execution for a set time, useful for allowing human review, system response, or staged containment actions.

A9:
Answer: D
Explanation: Targeted prompts ensure that critical decisions are routed to the right analysts, improving accuracy, accountability, and response quality.

A10:
Answer: C
Explanation: If a Decision Block does not match any condition and lacks a defined default path, the playbook can halt unexpectedly, which could impact incident response.

Formatted Output and Data Access Practice Question
A1:

Answer: B
Explanation: A Format Block creates clear, readable output by combining static text with dynamic variables pulled from previous blocks, such as artifact values or action results.

A2:
Answer: B
Explanation: This variable syntax accesses the destination address field from all artifacts within a container, making the output dynamic and specific to the event.

A3:
Answer: D
Explanation: Format Blocks are commonly used to create structured summaries that can be logged, emailed, or sent to collaboration tools such as Slack or MS Teams.

A4:
Answer: C
Explanation: Multiline formatting helps analysts understand incidents at a glance by clearly displaying event metadata like severity, IP, and affected users in a readable structure.

A5:
Answer: A
Explanation: `phantom.collect2()` is a built-in SOAR function used to extract specific values from data sources like artifacts or results of previous actions for further use.

A6:
Answer: B
Explanation: `phantom.debug()` is used to print messages and variables to the system logs for debugging and testing playbook logic during development.

A7:
Answer: D
Explanation: Data from action results or collected artifacts can be accessed using standard Python syntax for dictionaries and lists to retrieve nested values like status, risk_score, etc.

A8:
Answer: D
Explanation: When prompts include formatted output, analysts can quickly understand the situation using well-organized summaries, improving decision-making accuracy.

A9:
Answer: C
Explanation: Format Blocks are often used to log decisions and actions in the case timeline, serving both documentation and compliance purposes.

A10:
Answer: A

Explanation: This code safely retrieves the `risk_score` value from the first data element returned in the action result using Python's `.get()` method.

Modular Playbook Development Practice Question
A1:
Answer: B
Explanation: The Playbook Block is used to embed a child playbook inside a parent playbook, allowing modular design where the child handles a specific task and returns data.

A2:
Answer: C
Explanation: A child playbook should be focused on a single task like enriching an IP or isolating a host. It's designed to be reusable and called from multiple parent workflows.

A3:
Answer: D
Explanation: Best practice is to use decision blocks after calling a child playbook to check status and decide whether to retry, escalate, or log the issue.

A4:
Answer: C
Explanation: Modular playbooks break complex workflows into smaller parts, making them easier to maintain, test, and scale across teams or use cases.

A5:
Answer: B
Explanation: Modular design promotes reuse, reduces duplication, and makes troubleshooting easier by separating tasks into maintainable units.

A6:
Answer: A
Explanation: A child playbook should do one thing well. Combining unrelated logic defeats the purpose of modularity.

A7:
Answer: D
Explanation: To maintain resilience, parent playbooks should always check the output or status of child playbooks and handle empty or failed returns gracefully.

A8:
Answer: C
Explanation: Structured JSON output allows parent playbooks to easily parse, filter, and reuse data from children, making logic clean and robust.

A9:
Answer: B

Explanation: Modular playbooks are ideal when logic is complex, needs reuse across workflows, or is developed and maintained by multiple analysts.

A10:
Answer: B
Explanation: Use clearly labeled inputs and outputs, and structure the data (e.g., JSON) for readability, reuse, and debugging. This keeps interfaces between playbooks clean.

Custom Lists and Data Routing Practice Question
A1:
Answer: C
Explanation: A custom list acts as a user-defined dataset that can be used in playbooks for comparisons, filtering, enrichment, or decision-making.

A2:
Answer: A
Explanation: Custom lists are created and managed from the UI path: Administration > Custom Lists, where users can name, edit, and populate the list manually or via CSV.

A3:
Answer: C
Explanation: A typical use case for a custom list is maintaining a whitelist of approved domains or IPs that should not trigger alerts or actions.

A4:
Answer: B
Explanation: Dynamic routing allows SOAR to make intelligent decisions at runtime, such as selecting different assets or teams based on the content of an event.

A5:
Answer: A
Explanation: By comparing incoming IPs to a list of internal addresses, SOAR avoids triggering alerts for trusted internal activity, thus improving detection accuracy.

A6:
Answer: B
Explanation: The `phantom.get_list()` function retrieves the contents of a custom list so you can programmatically check for the presence of a value.

A7:
Answer: B
Explanation: Routing rules are created in ingestion settings and case templates, where logic defines which playbook to run, who to assign, or which tags to apply.

A8:
Answer: D

Explanation: Labels are predefined categories like “email alert” or “endpoint alert” that help SOAR determine routing paths and select appropriate playbooks.

A9:
Answer: C
Explanation: Tags provide lightweight, freeform metadata that analysts can use to label containers or artifacts, enabling easier filtering and routing logic.

A10:
Answer: C
Explanation: Multi-column custom lists allow more structured data, such as IP + reason or domain + confidence level, enabling richer logic within playbooks.

Configuring External Splunk Search Practice Question
A1:
Answer: C
Explanation: Integrating with Splunk allows SOAR to execute SPL (Search Processing Language) queries that return logs, alerts, or contextual data to enrich and automate security workflows.

A2:
Answer: C
Explanation: Before creating a Splunk asset, you must install the official Splunk app from the SOAR App Store, which provides the necessary action blocks for querying Splunk.

A3:
Answer: B
Explanation: If the connection fails, SOAR displays a connection error. The administrator must check network access, credentials, ports, and Splunk user permissions.

A4:
Answer: B
Explanation: The “Query Timeout” field controls how long SOAR waits for Splunk to return results for a query before timing out and marking the action as failed.

A5:
Answer: D
Explanation: The `head` SPL command is used to limit the number of returned events. `| head 100` limits the result set to 100 rows.

A6:
Answer: B
Explanation: Filters like time range and index scope reduce the volume of data returned, improving playbook speed and reliability while minimizing unnecessary load.

A7:
Answer: A

Explanation: A Splunk asset can authenticate using either a username/password or an API token associated with a user who has permission to run SPL searches.

A8:
Answer: C
Explanation: If the port is incorrect, SOAR will fail to establish a connection with Splunk during testing, resulting in a timeout or connection refusal.

A9:
Answer: A
Explanation: Using a dedicated service account improves security and auditability while avoiding disruption if a personal user account is changed or disabled.

A10:
Answer: C
Explanation: Running Splunk searches from SOAR enables dynamic enrichment, such as retrieving related logs, recent activity, or historical patterns tied to an IP, host, or user.

Integrating SOAR into Splunk Practice Question
A1:
Answer: A
Explanation: The Splunk App for SOAR enables the forwarding of alerts to SOAR and provides dashboards in Splunk that allow users to monitor SOAR cases and automation activities from within Splunk.

A2:
Answer: B
Explanation: ARF allows analysts in Splunk Enterprise Security to trigger response actions in SOAR (e.g., run a playbook, escalate an event) directly from search results or notable events.

A3:
Answer: C
Explanation: Correlation searches configured with a custom alert action can send event data to SOAR, initiating the automation process.

A4:
Answer: D
Explanation: SOAR receives data via REST API calls that originate from Splunk alert actions or ARF, using a configured Splunk asset in SOAR.

A5:
Answer: B
Explanation: Playbooks triggered by Splunk events can enrich data, run automation tasks (e.g., block IPs, disable accounts), and provide updates back to Splunk.

A6:
Answer: C
Explanation: The integration allows analysts to see SOAR case progress and automation outcomes directly in Splunk dashboards, improving workflow continuity.

A7:
Answer: D
Explanation: Incorrect configurations, such as bad API credentials or endpoint URLs, are common causes of failed alert delivery from Splunk to SOAR.

A8:
Answer: C
Explanation: SOAR can use data from threat intelligence and enrichment sources to validate and escalate Splunk alerts more intelligently.

A9:
Answer: B
Explanation: Real-time correlation searches paired with custom alert actions can push events to SOAR instantly, minimizing delays.

A10:
Answer: A
Explanation: Alert actions should contain actionable information such as IPs, usernames, and other indicators needed by SOAR playbooks to investigate and respond.

Custom Coding Practice Question
A1:
Answer: A
Explanation: Custom code allows SOAR users to go beyond what the visual editor provides by using Python scripts for complex logic, data parsing, and custom system integrations.

A2:
Answer: B
Explanation: Custom Functions are Python-based units of reusable logic that can be shared and reused across multiple playbooks to perform specific tasks or data manipulations.

A3:
Answer: B
Explanation: Code blocks in playbooks are used for fine-grained control such as iterating through data, applying filters, or handling complex formatting or logic operations.

A4:
Answer: C
Explanation: The `metadata.yml` file defines all the metadata for a custom app in SOAR, including action names, descriptions, input fields, and data types.

A5:
Answer: A
Explanation: The `requests` library is widely used in custom code for making outbound API requests to third-party systems, especially when native app actions are not available.

A6:
Answer: D
Explanation: Custom Functions should return structured output such as a dictionary or JSON, which allows for clean and consistent data passing between blocks or playbooks.

A7:
Answer: B
Explanation: A Custom Function is ideal for reusable data transformation, such as extracting TLDs from domains. It encapsulates logic that's too complex or repetitive for visual blocks.

A8:
Answer: B
Explanation: `phantom.debug()` prints output to the playbook run log, which is helpful during development and troubleshooting of custom code blocks or functions.

A9:
Answer: C
Explanation: The action handler script in a custom app contains the Python code that defines what happens when an action is triggered by a playbook or user.

A10:
Answer: C
Explanation: Sanitizing and validating input helps protect against injection attacks or logic errors, especially when input comes from external or untrusted sources.

Using REST Practice Question
A1:
Answer: A
Explanation: The REST API enables external systems and scripts to create containers, trigger playbooks, retrieve artifacts, and more — providing remote control over SOAR operations.

A2:
Answer: D
Explanation: Token-based authentication is the most secure and scalable method for interacting with the API, as it avoids exposing passwords and can be scoped or rotated easily.

A3:
Answer: D
Explanation: This endpoint is used to programmatically create a new container (event) within SOAR — commonly used when pushing alerts from other tools.

A4:
Answer: A
Explanation: The Python `requests` library provides high-level functions like `get()` and `post()` which are commonly used to interact with REST APIs.

A5:
Answer: A
Explanation: Over-permissive tokens increase the risk of unauthorized or accidental misuse of the API, such as deleting containers or leaking sensitive data.

A6:
Answer: D
Explanation: The `/rest/playbook_run` endpoint is used to manually trigger the execution of a specified playbook — useful for integrations with ticketing or workflow systems.

A7:
Answer: A
Explanation: HTTPS protects sensitive data (tokens, credentials, log details) from interception during transit. Using plain HTTP introduces serious security risks.

A8:
Answer: B
Explanation: `phantom.debug()` allows developers to output logs inside playbooks, useful for inspecting variables and troubleshooting logic during development.

A9:
Answer: A
Explanation: REST API integration allows external systems like Jira, ServiceNow, or detection tools to create SOAR events, trigger responses, and retrieve results programmatically.

A10:
Answer: B
Explanation: Postman is a GUI-based tool used for testing and documenting REST APIs. It helps explore endpoints and build integration workflows.