The bridge to possible

Designing Security Automation Playbooks - Sharing Lessons Learned with Practitioners

Contents

The bridge to possible

Abstract

Security automation playbooks are a guide for responding to security incidents, allowing security teams to develop and deploy automatic processes. They are critical to achieving security outcomes in a timely manner. However, many enterprises lack effective automation processes and playbooks in their Security Operation Center (SOC) portfolio. A playbook needs a well-thought-out design to be effective.

With proper design, you can quickly build a playbook and realize the value of automation in your security operation.

In this white paper, we share challenges in designing playbooks, good practices, and concrete examples from existing deployments to help you drive successful operations with automation.

Introduction

Security Orchestration, Automation, and Response (SOAR) is a technology platform that automates security operation tasks by deploying and executing playbooks. Realizing the value of a playbook is not the most straightforward task; it takes practice, time, and resources. If you are unfamiliar with security automation and SOAR, then this white paper can be a good starting reference.

In this series of white papers, we share our experience gained from the extensive automation work we have carried out for multiple large SOCs, including designing and implementing automation frameworks and effective playbooks. This white paper is part 1 of a trilogy. In part 1, we share best practices for designing security automation playbooks.

Let's start by highlighting core principles that can drive your approach to design an automation framework and playbooks.

- **Look at the big picture:** Automation is not about focusing on the individual tasks and steps; but it is about analyzing processes and finding ways to reduce manual interventions. Typical security automation consists of multiple tasks and focuses on end-to-end processes.
- **Never design monolithic playbooks:** Like software code, monolithic programs are hard to build and your worst nightmare to maintain and operate. If you think about it, automation playbooks are nothing but code!
- **It is beyond automating containment:** Think about triage, false positive management, enrichment, ticketing, KPI management, reporting, notification, and so on. These are areas that you can, and in many cases should, automate.
- **Document and standardize:** Follow a standard template when you document your playbooks and ensure that enough information is added to your playbooks when deployed on your SOAR of choice.
- **Don't just rely on what comes out of the box:** We look at SOAR as a platform that allows you to program your specific requirements (e.g., develop new integrations or create custom code using a programming language such as Python).
- **You don't have to automate everything:** Consider the return on investment when deciding to design and deploy playbooks. In some cases, automation brings minimal return compared to the time and effort you invest in designing, deploying, and operating automation!

The guidelines in this document are platform independent, although they come primarily from working on large projects involving Cisco SecureX and Splunk Phantom.

Challenges in designing a playbook

When embarking on a security automation journey, you will face challenges that span processes, people, and technology. The following are challenges that we typically see organizations facing.

- **Lack of workflows.** In many cases, you may not have a workflow or process to help you start designing a playbook. You will need to figure out and understand the details involved, like peeling an onion. Identifying and documenting the workflow will drive other activities, such as identifying what would trigger the playbook, understanding existing technologies, and discussing how teams respond to threats and security alerts.
- **The requirement of a playbook is constantly changing.** It is a common challenge for end users to frequently alter their requirements during playbook development as requirements evolve through ongoing discussion and testing. If you thoroughly examine your use cases before jumping to the canvas, then your automation playbook code will cover most of the requirements. Don't hard code things; reusable is a golden rule.
- **Data is not available.** The playbook will work as designed when input data is complete and available. In practice, this input data may not be available, may be incomplete, or might require further processing. For example, if a playbook is designed to notify an end user that his/her account was disabled in response to a security event and that event did not contain the user's email address, you must reach out to other systems, such as Windows Active Directory, to retrieve the user's email address.
- **Integration.** In most cases, built-in integrations and playbooks in a SOAR platform would not be sufficient to meet your requirements. For example, when containing an endpoint using Cisco Secure Endpoint (formerly known as AMP for Endpoints), the built-in application in Splunk Phantom can provide very limited integration features. You can support more features by developing a new application or connector on your SOAR, such as identifying if you have already quarantined the endpoint before executing a containment action, a necessary prequarantine step to take.

Playbook design consideration

A successful system follows solid design principles. Playbook design consists of workflow preparation, planning, layout, and documentation. The playbook principles in Figure 1 apply to use cases deployed in SOC.

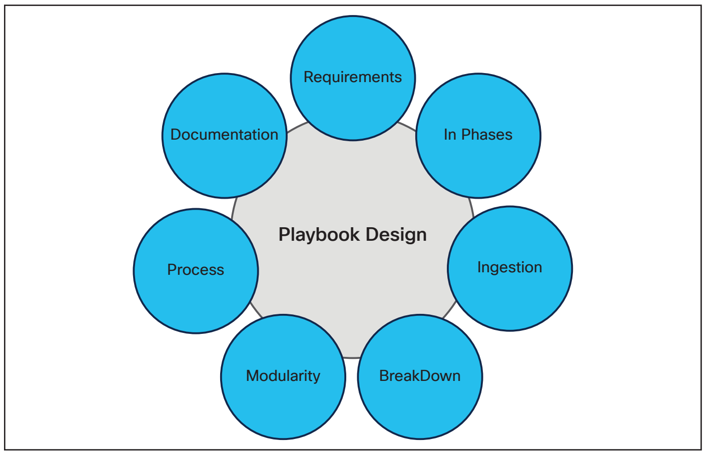


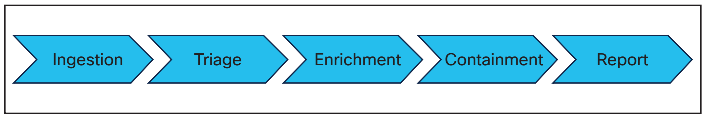
Figure 1. Playbook design principles

Knowing playbook requirements. Before you start designing a playbook, you need to know the requirements and expected outcomes. There are a few key questions that can help you identify the requirements of a playbook[1]:

- Will it run this workflow on a schedule, triggered by an event or manually?
- What is the triggering condition for the playbook (e.g., email, security event, API call, etc.)?
- What data will you receive or collect, and in what format (e.g., JSON, XML, pure text, html, etc.)?
- Is the data ready to be processed by the playbook, or do you need to parse it further?
- Are there existing incident response plans and processes available?
- Do you have the required integrations and tools in your SOAR?

You may not have all the answers at the beginning of the playbook design, but you could follow a common architecture pattern across all your playbooks to base your design. Playbook design is iterative as additional information becomes available.

In phases. When designing a playbook, think about the phases that an event or an incident will go through. This approach will help to keep your playbook modular. Each phase should have expected outputs for the next phase in line to process. The typical phases are **ingestion, triage, enrichment, containment,** and **reporting**[2].



- **Ingestion (Triggering):** Ingestion can be based on a schedule, event, email, or ticket depending on your workflow.
- **Triage:** Identify false positive events and classify the event.
- **Enrichment:** Reach out to other services for context.
- **Containment:** Take response action, when required, such as blocking an IP address on the firewall.
- **Report:** What the playbook did and what is the result.

Each of these phases may have a corresponding process followed by the operation teams. For example, containment is an essential component in many playbooks. Firstly, you need to identify the criteria for taking an action and its criticality. Common criteria are malicious Indicators of Compromise (IOC) such as IP addresses, domains, URLs, file hashes, and so on. The next step is to decide what system(s) you want use to perform the containment action on, as there can be various approaches to achieve that goal, such as host-based isolation, or IP/MAC address block.

Ingestion. Injection sources (triggering) can be various such as webhook events, emails, or security events from Security Information and Event Management (SIEM). Make sure what data you should ingest to augment analysis. Always choose the right place to process or normalize data, for example, Splunk Enterprise Security (Splunk ES) can carry and map necessary data or fields and send them to Cisco SecureX. It is more efficient to send these fields by Splunk ES rather than by enriching data on Cisco SecureX.

Ingestion/triggers are one of the key inputs of the playbook. You need to understand the scenario of triggers. There are a few questions that can help to identify triggers[3]. How do I detect this incident, and what analysis data would be available for this trigger? For example, a malware event can be the trigger event for the malware playbook. You need to list all input and output key fields in a detailed playbook design.

Breakdown by module. When there is architecture or high-level workflow, most of the time you will break it into subflows starting from top to bottom. Each subflow may need multiple iterations.

[Example – Containment playbook]

- **Requirement** – The containment playbook aims to contain/quarantine a specific infected host using IP/MAC address. Internal IP containment can be done immediately. External IP can be done at a scheduled (nonpeak) time to avoid end-user impact, keeping each firewall schedule in mind.
- **Design** – Internal IP can be blocked on Cisco Secure Endpoint if an associated host has registered. “If not found” blocking can be on Identity Service Engine (ISE). External IP can be blocked on perimeter firewalls. The approval playbook can also be invoked before taking containment action.

Below is part of the containment playbook workflow for showing the high-level and breakdown process for the internal IP/MAC address block.

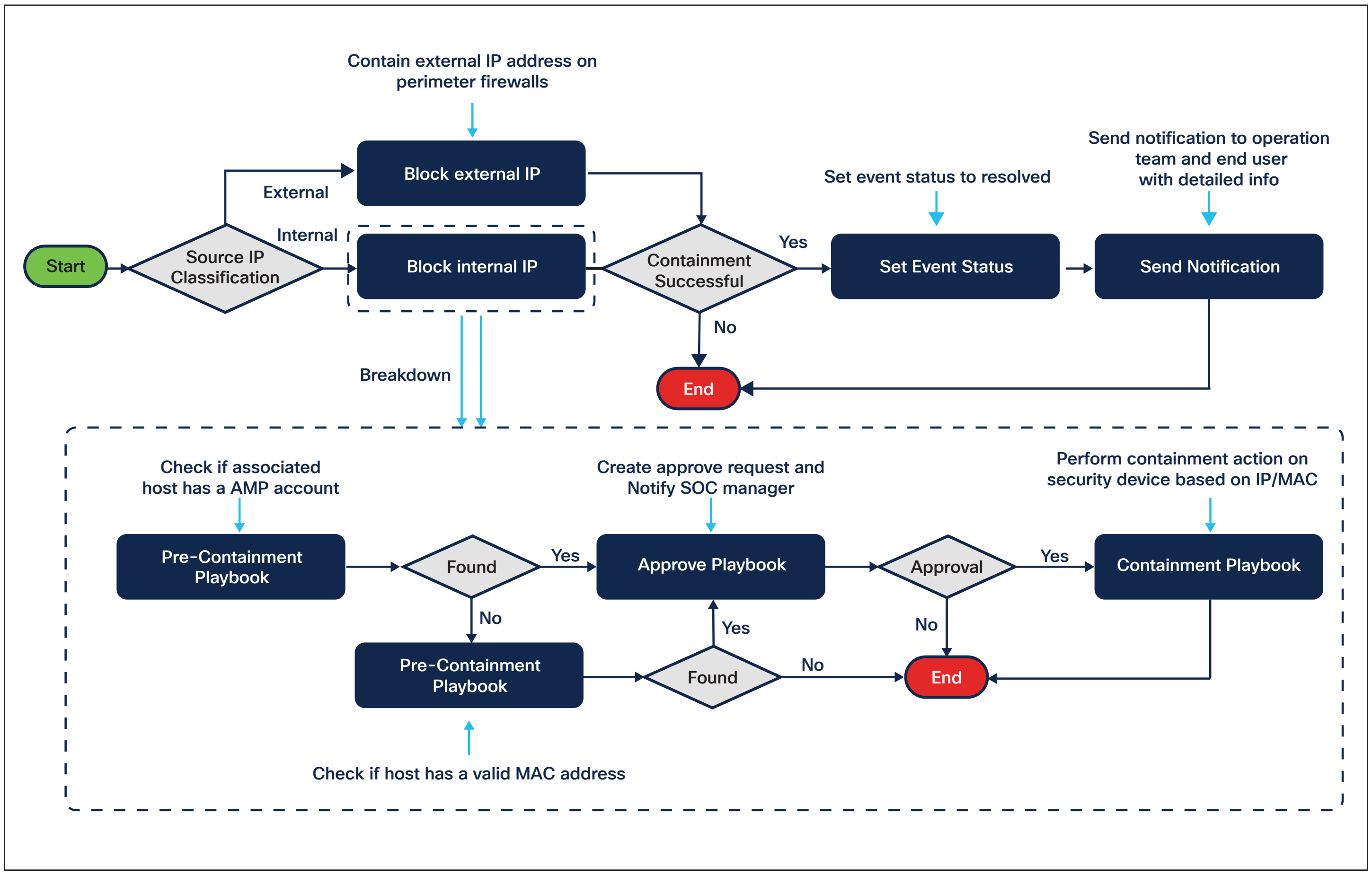

Contain external IP address on perimeter firewalls
Block external IP
External
Internal
Start
Source IP Classification
Block internal IP
Containment Successful
Yes
No
Set event status to resolved
Set Event Status
Send notification to operation team and end user with detailed info
Send Notification
End
Breakdown
Check if associated host has a AMP account
Pre-Containment Playbook
Found
Yes
No
Create approve request and Notify SOC manager
Approve Playbook
Approval
Yes
No
Perform containment action on security device based on IP/MAC
Containment Playbook
Pre-Containment Playbook
Found
Yes
No
End
Check if host has a valid MAC address

Figure 2. Containment workflow and breakdown

Modularity. Focus on making code reusable during playbook design and implementation phase. Focus on abstraction during implementation. Also, try to leverage modular or atomic features of your SOAR platform.

In Cisco SecureX, an atomic playbook should pivot on one core function. Leverage subworkflow instead of an atomic playbook for variations and additional functionality. It is recommended to set the target at the beginning of the workflow and add a detailed description to the atomic playbook if you plan to use one.

In Splunk Phantom, you can use app actions or custom functions to achieve basic modular functions. If you are doing a little more complicated logic or more than a couple of things, then make it an atomic or subplaybook.

What is the excellent process or destructive process in design? A playbook will process data based on a predefined flow or function. A good playbook will heavily rely on a solid process, whether a simple or complicated, sequential, or parallel order. You will need to consider all possible scenarios.

In most cases, it may not be a good process if all steps are designed sequentially as it may not cover all cases. You need to set some checkpoints or measurement points to make sure the necessary data is valid. You also need to introduce decision points to match different situations and a decision point to identify if the performed action is a success or not. It may go to an additional flow based on different results.

- **Parallelism** – It is recommended to make use of multiple threads, either via parallel blocks or asynchronous processing, whenever you can. In the meantime, consider the ability of them to be independent of each other's outputs and make the thread safe. This will speed up execution time.
- **Design for different scenarios** – Consider and address all possible flows or exceptions that a playbook might go through[3]. For example, when inserting an approval step, the person approving might accept or reject the request; it is also possible that the approval times out before that person takes action. The combinations will lead to different subsequent flows.

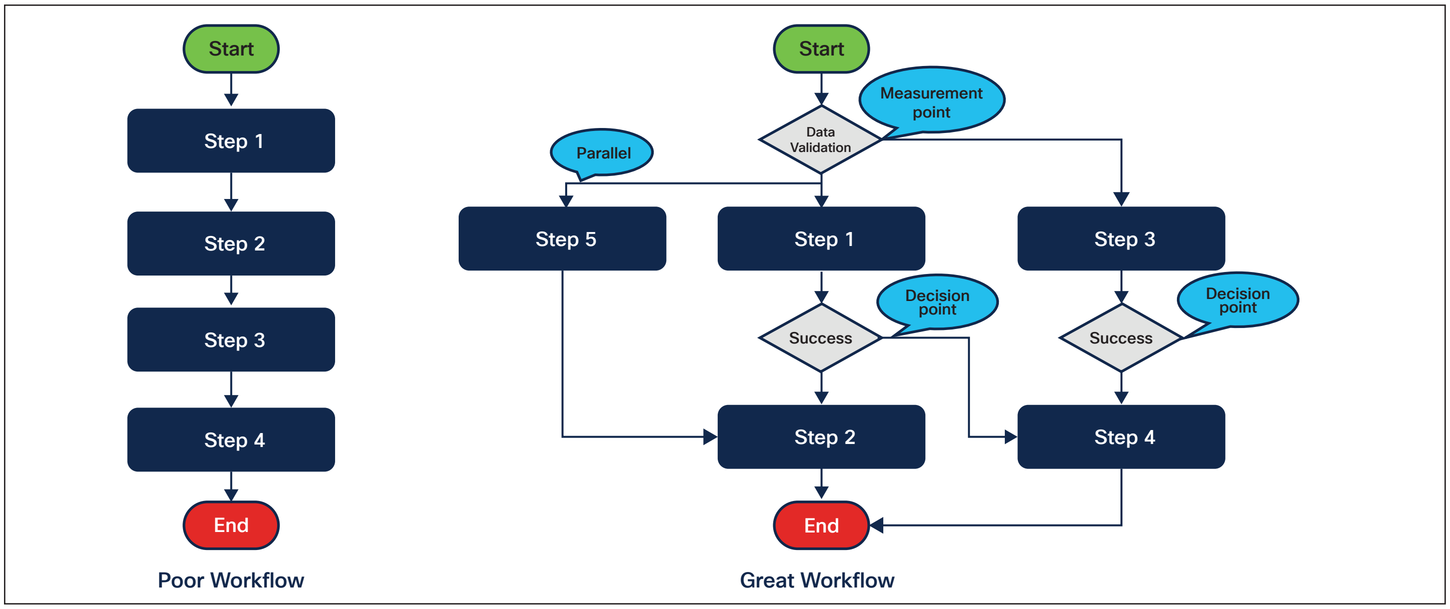


Figure 3. Workflow comparison

Documentation. After you capture all essential information, you can put all key info into the table below and prepare a workflow diagram.

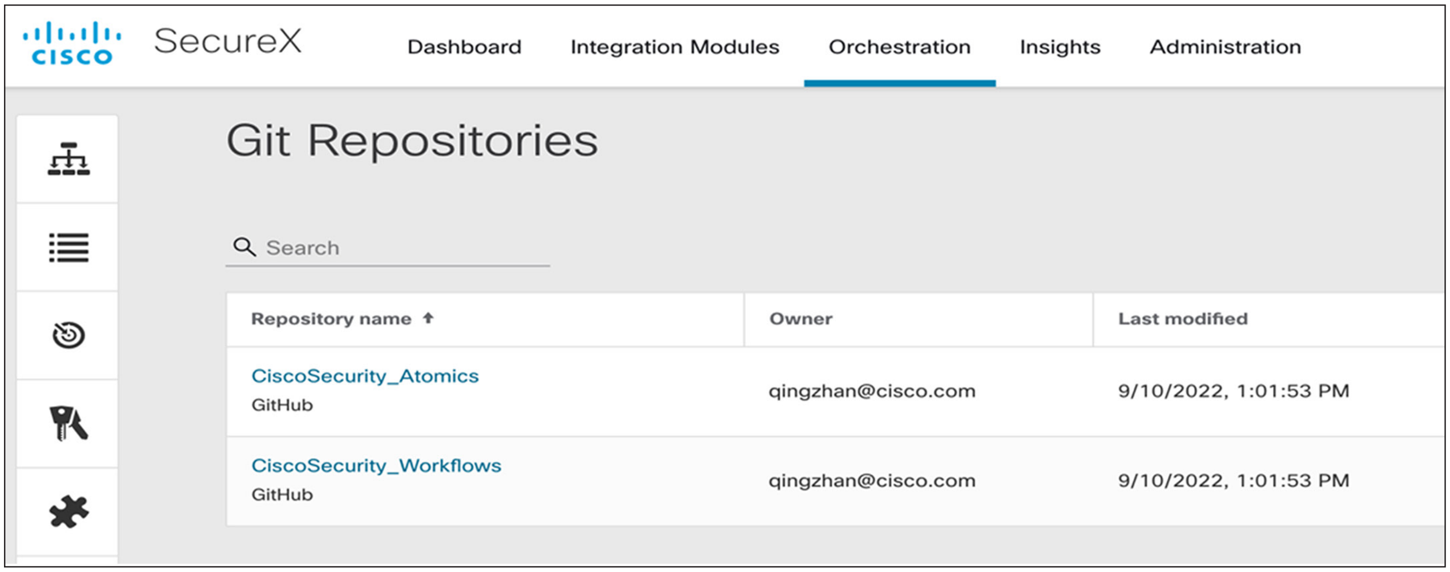
Table 1. Playbook document template

Item	Description
Reference Number	PLAYBOOK-Name
Status	Draft
Description	The playbook covers the automation of tasks related to the triage, investigation, and containment of an alert. Notable events generated by Splunk ES would be based on the Malware or Intrusion Detection data models.
Integration	Cisco Firepower, Cisco Secure Endpoint, Cisco ISE
Called Playbook	Atomic Playbook (notification, containment)
Repositories	Cisco CX Repo
Active Setting	Notable event or scheduled
Trigger Condition	Triggered on receiving an alert generated by the Splunk with label
Input Data: Type	Source IP: String, Destination IP: String, Hostname: String, Signature: String, Filehash: String
Output Data: Type	Indicator_malicious: String, Event Status: String, Containment: String

Consider a top-down approach while drawing a playbook flow. This flow will reflect your security design. You can then build detailed flows that can be reiterated and optimized. You would typically conduct iterations of refinements and updates as you get customer feedback.

Playbook tip:

- **Sketch** – Draw on paper before jumping to a digital canvas as the start. Write a "pseudo" code or write out how I would do something manually. Do it in a workflow "style," and then it is much easier to translate that to a digital workflow.
- **Naming convention** – Make naming consistent and easy to understand for variables, activities, and workflows. Avoid using the system default names as they don't provide information about the workflow.
- **Export and versioning** – Using Git repositories for export/import and version management is recommended. Leverage cloud-based repositories as it reduces maintenance overhead. Most SOAR can support external repositories. Below is an example of Cisco SecureX repository integration.



Git Repositories

Repository name	Owner	Last modified
CiscoSecurity_Atomics GitHub	qingzhan@cisco.com	9/10/2022, 1:01:53 PM
CiscoSecurity_Workflows GitHub	qingzhan@cisco.com	9/10/2022, 1:01:53 PM

Figure 4. Git repositories

Example – Triage playbook

- **Requirement** – The triage playbook aims to identify whether incoming events are false positives or not.
- **Analysis** – In the triage playbook, the triggering event is a notable event predefined in Splunk. To identify false positives, the playbook needs to run a search against Splunk with two key conditions: event signature and if similar events were marked as false positives historically. While it is crucial to choose a field as a signature, it could vary based on the type of event. For example, malware name can be a signature. Make sure the required fields are carried into the notable event.
- **Triggering** – Notable event with a label that is predefined in Splunk
- **Input** – Hostname, IP, username, signature, hash
- **Output** – False positive flag, event status
- **Design** – The playbook will run due to incoming events (triggering). Search previous notable events received in the last x days and auto-close these notable events if similar events were tagged as false positives by analysts.

The triage process is a key capability to sort out false-positive events. The diagram below shows a customer security event filtered from 200+ to 20+ with the triage playbook running.

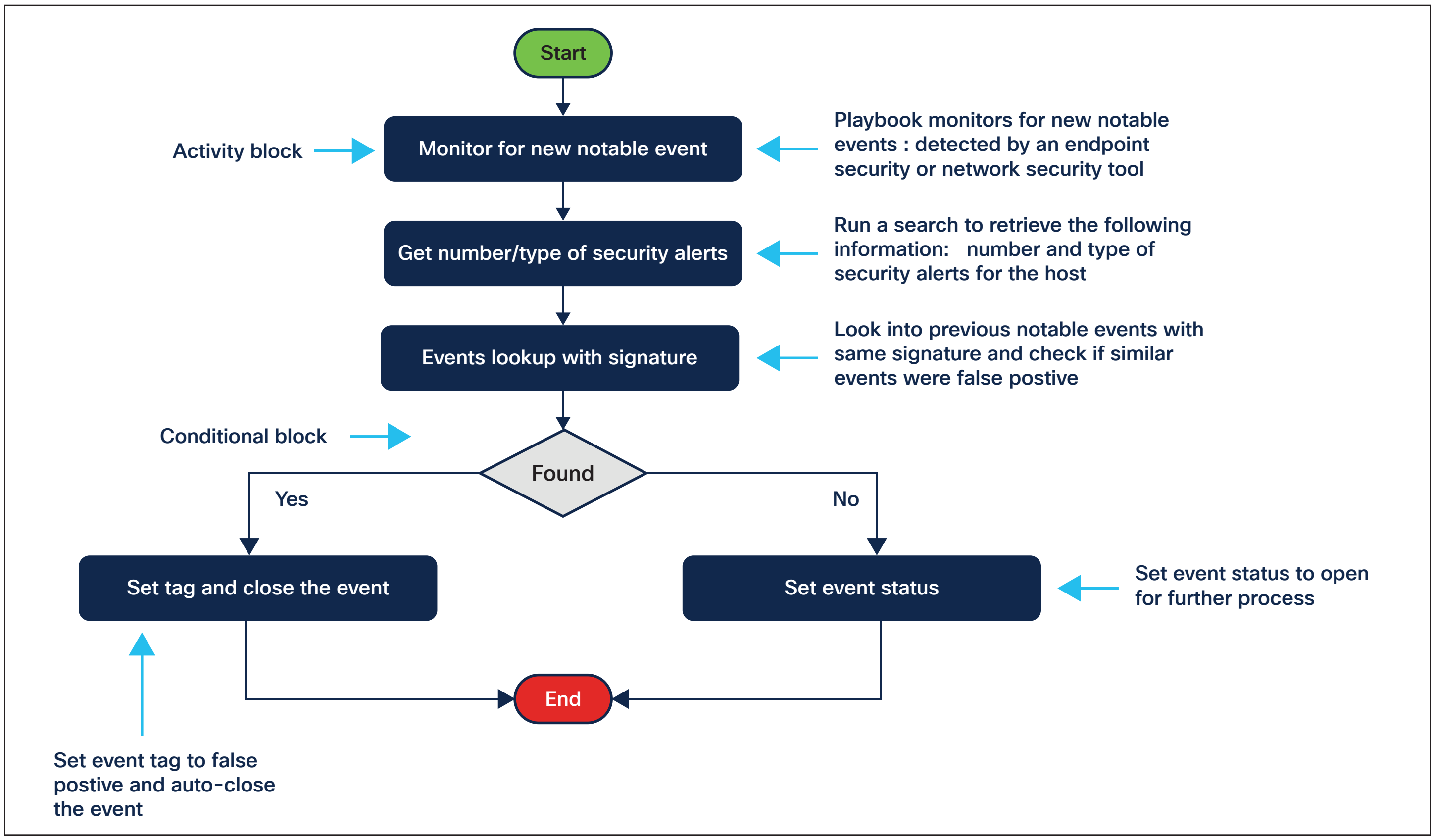

Start
Activity block
Monitor for new notable event
Playbook monitors for new notable events : detected by an endpoint security or network security tool
Get number/type of security alerts
Run a search to retrieve the following information: number and type of security alerts for the host
Events lookup with signature
Look into previous notable events with same signature and check if similar events were false postive
Conditional block
Found
Yes
No
Set tag and close the event
Set event status
Set event status to open for further process
End
Set event tag to false postive and auto-close the event

Figure 5. Triage playbook workflow

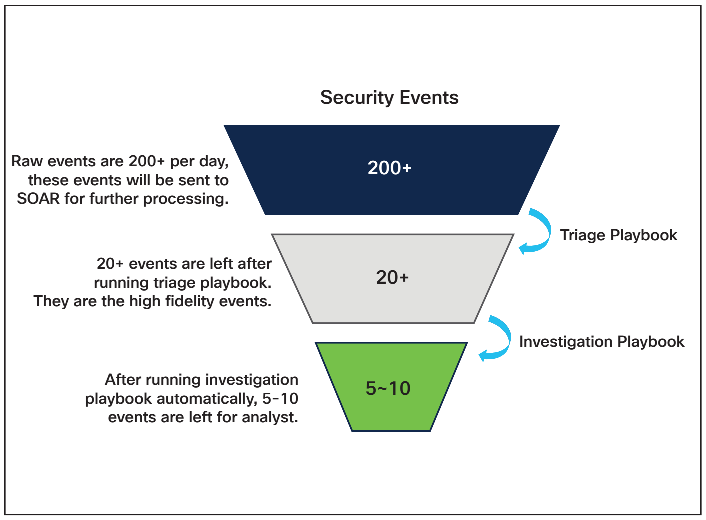

Security Events
200+
Raw events are 200+ per day, these events will be sent to SOAR for further processing.
Triage Playbook
20+
20+ events are left after running triage playbook. They are the high fidelity events.
Investigation Playbook
5~10
After running investigation playbook automatically, 5-10 events are left for analyst.

Figure 6. Events volume with triage playbook

Conclusion

Building and maintaining automation is a continuous process. Start somewhere and build from there. Keep a compressed design with outcomes in mind. A good starting point would be identifying repetitive manual tasks that can be automated.

End-to-end security automation takes time, mainly due to the large number of interdependencies. These interdependencies of processes pose major challenges. Below are a few best practices.

- Triage and enrichment could be relatively more straightforward to implement than containment, and it is recommended to achieve triage and enrichment first.
- Try to use automation to replace some manual tasks first, as there is an existing process to follow.
- Start with enrichment automation in Phase I and proceed with response automation in Phase II.
- SOC orchestration and automation need security and programming/scripting expertise.

Authors

Qingguo Zhang, Technical Leader, Cisco Systems Inc

Reviewers

Nadhem AlFardan, Principal Architect, Cisco Systems Inc

Frederic Detienne, Distinguished Engineer, Cisco Systems Inc

Jay Johnston, Principal Engineer, Cisco Systems Inc

Tim Rowley, Principal Architect, Cisco Systems Inc

Hemal Surti, Principal Architect, Cisco Systems Inc

Shaun Roberts, Principal Engineer, Cisco Systems Inc

Donal Gallagher, Consulting Engineer, Cisco Systems Inc

Reference

[1] Workflow Best Practices - SecureX Readiness Training - Shaun Roberts

[2] Splunk Conf Slides

Pushing a Phantom - Playbook Architecture To Get the Team to Actually (Splunk) SOAR SEC1266B - Greg Rivas - Splunk.conf22

Our Splunk Phantom Journey - SEC1506 - Splunk.conf22

[3] Security Operations Center Lessons from the Trenches - Hassan Mourad - D3C4S1_-_SOC_-_The_Story_VT_1_0_Sharable.pptx