Optimal Automated Generation of Playbooks

Kéren A Saint-Hilaire, Christopher Neal, Frédéric Cuppens, Nora Boulahia-Cuppens, Makhlouf Hadji

Optimal Automated Generation of Playbooks

Kéren A Saint-Hilaire[1,2][0009−0003−8139−3992], Christopher Neal[1,2][0000−0002−6953−8728], Frédéric Cuppens[1][0000−0003−1124−2200], Nora Boulahia-Cuppens[1][0000−0001−8792−0413], and Makhlouf Hadji[2][0000−0003−1048−753X]

[1] Polytechnique Montréal Montréal, Canada
{keren-a.saint-hilaire,christopher.neal,frederic.cuppens,
nora.boulahia-cuppens}@polymtl.ca
[2] IRT SystemX Palaiseau, France
{keren.saint-hilaire,christopher.neal,makhlouf.hadji}@irt-systemx.fr

Abstract. Cyberattacks have become more complex and analysts need help managing all alerts promptly. Many organizations implement Security, Orchestration, Automation, and Response (SOAR) tools to automate Incident Response (IR). However, it is challenging to integrate these tools, often delaying expected Return on Investment (ROI). Our approach aims to automatically generate optimal playbooks using the Pareto front, which balances impact, loss, and complexity. These playbooks are populated in an ontology that aims to be integrated with a SOAR to overcome the SOAR limitations. Using the Pareto Front, we aim to reduce the generated playbooks by an average of over 75%.

Keywords: Network security · Playbooks · Security management

1 Introduction

The global increase in cyberattacks concerns all organizations using IT components. Organizations monitor network traffic and information systems to identify potential attacks and trigger security alerts for investigation. However, the Computer Security Incident Response Teams (CSRIT) are overcome by the volume of incidents they must analyze manually. Analysts face too many alerts, leading to delayed response or desensitization to the alerts.

To ease analysts' work, Security, Orchestration, Automation, and Response (SOAR) systems help by ingesting data from sensors and Security Information and Event Management (SIEM) systems to automate responses. However, SOAR integration is complex and time-consuming, requiring the integration of all security tools. It also demands severe financial investment, such as buying tools and engaging specialized personnel. Therefore, organizations often see a delayed Return on Investment (ROI) and only automate small tasks.

There is a lack of research concerning the SOAR, while no standards exist for its interoperability. Playbooks, which define how to execute automated actions, vary in format and lack formalization, so responses are not uniform

for the analyst. We propose using an ontology to formalize Incident Response (IR) playbooks to address SOAR limitations. To enhance interoperability, we propose automatically generating optimal playbooks, balancing conflicting objectives, and integrating them into an IR playbook ontology for use in any SOAR system.

We base our approach on countermeasures selected from matching the MITRE D3FEND KG[3] and the Vulnerability Description Ontology (VDO) KG [1]. The playbooks are automatically generated, matching the selected countermeasures with the IR actions from the RE&CT framework[4].

We identify only the optimal playbooks thanks to the Pareto front [12] and choose from any of the optimal playbooks. We populate the IR playbook ontology with instances for the selected playbook, then validate the approach through a use-case scenario. We evaluate the approach by comparing the gap of playbooks generated before and after applying Pareto and based on the time cost.

2 Related Work

The National Institute of Standards and Technology (NIST) describes IR as an action plan an organization can follow to contain an attack and recover from it [2]. According to NIST, the lifecycle of an IR includes the steps of Preparation, Detection & Analysis, Containment, Eradication & Recovery, and Post-incident activities.

The RE&CT framework inspired by MITRE ATT&CK [5] is a knowledge base of attack techniques and tactics that allows for categorizing IR techniques and actions. However, dependencies and automation amongst actions are not explained within RE&CT. Therefore, we define playbooks using our IR ontology and provide more information for executing actions. Oasis's Collaborative Automated Course of Action Operations (CACAO) playbook schema offers structured and standardized playbooks with logical steps and workflow definitions, enhancing automation potential [7] that we use in our approach.

Automated playbook generation varies. Empl et al. [3] extract CACAO playbooks from CSAF documents, but limitations arise when documentation lacks asset information. We propose a knowledge-based approach to fill this gap. Sarda et al. [8] use Large Language Models (LLMs) for anomaly detection and remediation. Islam et al. [6] automate security tool integration with NLP models and an ontology. Mern et al. [9] propose an Automated Cyber Security Orchestrator (ACSO) using Reinforcement Learning (RL), which can be time-consuming. Our approach generates playbooks by aligning knowledge bases to populate a playbook ontology for SOAR integration.

Moreira et al. [10] advocate using an ontology for security incident management, defining critical concepts like action and event, also found in the Unified Cybersecurity Ontology (UCO) [11]. Hutschenreuter et al. [5] apply ontologies

[3] https://d3fend.mitre.org/
[4] https://atc-project.github.io/atc-react/
[5] https://attack.mitre.org/

to port infrastructure resilience, introducing concepts like threat and security measures. While these ontologies offer relevant concepts, adaptation is needed. We employ an ontology to formalize IR playbooks for SOAR integration. We reuse some concepts proposed in [5, 10] in our ontology.

Ganin et al. [4] introduce a multicriteria framework for risk management. Our approach similarly considers multiple criteria, utilizing the Pareto front to select dominant playbooks based on impact, complexity, and loss. We choose these parameters based on knowledge from the literature review. We also rationally fix their values based on state-of-the-art knowledge. The Pareto front prioritizes dominant playbooks regardless of specific criteria values, ensuring fairness in selection.

3 Our approach

3.1 The Proposed Approach

Figure 1 illustrates our optimal IR playbook generation process. Countermeasures are selected through KG matching, obtaining a list categorized by the RE&CT framework's IR phases. Our IR playbook ontology provides attributes for the IR actions, such as required tools, complexity, loss, and impact values. The list of actions is pruned, matching these attributes with system and attack details. All possible playbooks are generated from the refined list, ensuring each playbook exceeds a set impact threshold. The Pareto front allows the identification of the optimal playbooks and the selection of one to create ontological individuals for our playbook ontology.

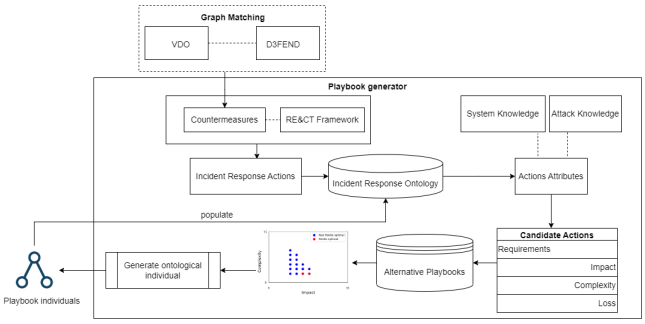

Fig. 1: Our Optimal Playbook Generation Process

3.2 IR playbook Ontology

This section describes our IR ontology[6] represented by Figure 2. We reuse some classes from existing ontologies. However, they are insufficient for modeling the

[6] https://github.com/phDimplKS/playbook_ontology

playbook concept. We create new classes to represent better and formalize the domain. The class **Playbook** is a super-class of **Playbook Step**, identifying the successive steps corresponding to IR actions; each individual of the playbook step class defines the action's position in the playbook. We take the class **Action** from the UCO [11] ontology. The classes **Tool** and **Security Tool**, taken from the existing ontologies UCO and SecOrp [6], represent assets involved in action completion. The class **Requirement**, created based on the requirements linked to each action from RE&CT, is equivalent with **Security Tool**, which is a subclass of **Tool**. Data represented by the class **Artifact** can also release actions. These actions released allow us to respond to attack tactics and techniques used by an attacker as described in the MITRE ATT&CK matrix; we represent these concepts by the class **ATTACK Thing**, which has the subclasses **Offensive Tactic** and **Offensive Technique**. The execution of an action requires releasing commands. Each **Command** individual is linked to a security tool and an action.

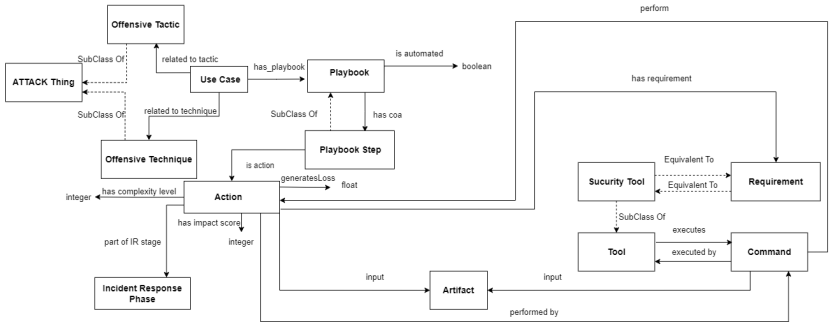

Fig. 2: Our IR Playbook Ontology.

The class **Use Case** concerns a specific use case monitored by an Intrusion Detection System (IDS). We also create our ontology properties representing the relationship between the individuals of the different classes. Each containment action is linked to a recovery action thanks to the property **linked to recovery action**. There is no concept in an ontology that allows different action sequences. We then create the following properties. The property **has coa** is directed to the first step of a playbook (*coa* signifies *Course of Action*). The property **has next** determines which playbook step should follow in the playbook workflow. Properties **has next if false** and **has next if true** allow representing the order of the actions when a condition exists. The property **parallel** expresses the fact that two actions can be executed at the same time.

We populate our ontology with individuals created from two existing security knowledge bases: 1) the MITRE ATT&CK framework constitutes a base for representing attack concepts, and 2) the RE&CT framework is the base for expressing IR. We instantiate the sub-classes **Offensive Technique** and **Offensive Tactic** with knowledge from MITRE ATT&CK. We do this population

process semi-automatically. We instantiate the class **Incident Response Phase** with the phases defined in RE&CT; the RE&CT actions allow instantiating the class **Action**. For the **Requirement** class, RE&CT proposes requirements for some actions, so we populate the ontology with them. However, no requirement is proposed for some actions, so we should consult security blogs to find their requirements and populate the ontology. We populate the classes **Playbook Step** and **Playbook** with the individuals created for the optimal playbooks selected.

3.3 Countermeasures Selection

This section details the automated process of selecting countermeasures for an exploited vulnerability using KG matching. The dashed part in Figure 1 illustrates the graph-matching process, with VDO KG and D3FEND KG as inputs. After parsing the KGs and conducting text processing to create corpora, Word2Vec models are trained, and KGs are embedded. Cosine similarity calculation matches attack impacts and attacker methods entities with their most similar offensive techniques in D3FEND. A query on D3FEND retrieves artifact entities linked to the matched offensive techniques.

When no artifact is linked to the offensive techniques, cosine similarity helps find the most similar artifact entity of D3FEND to its corresponding entity in VDO. Finally, our solution automatically queries the defensive techniques linked to this artifact from D3FEND, the countermeasures against the vulnerability exploited. Each countermeasure is automatically linked with an IR phase and a category of the RE&CT framework. The candidate countermeasures selected serve as input to our optimal playbook generation solution.

3.4 Optimal Playbooks Generation

As Figure 1 shows, the playbook generator first generates a list of IR actions, correlating the countermeasures with the RE&CT framework. Our solution queries the action attributes for its release. The playbook generator verifies that the system contains at least one required security tool for the action execution. In addition, it verifies the needed attacker's position to release the attack. For a remote attack, it discards the action *block internal IP address*.

The playbook generator generates different combinations of actions considering the following conditions. We fix a threshold value that the sum of all actions' impact values must exceed. A playbook should contain at least 1 action from two different phases: Identification, Preparation, Containment, and Eradication phases, considering that an action of the identification phase is necessary to detect and analyze an anomaly, and then at least one action from the containment or eradication phase should be taken to mitigate the threat detected.

We fix a maximum number of actions in a playbook to reduce the execution time required to generate a playbook. Based on our experiments presented in Section 4.1, we notice that we get an average of more than 20 alternative playbooks for a vulnerability. In order to choose the optimal one, we use the concept of Pareto optimality in our process. The optimality objectives are: **a)**

maximize the impact, **b)** minimize the complexity, and **c)** minimize the loss. All the playbooks that have no other playbooks dominating them are Pareto optimal.

The recovery action for each containment action in the optimal playbooks and the lessons learned phase's actions are added automatically to these playbooks. One of the optimal playbooks can be selected randomly, since they are all equally optimal. Our approach automatically generates individuals for the chosen playbook to populate our playbook ontology. An instance for the playbook class is created, linked with the first playbook step through the **has coa** property. Playbook steps are associated using the **has next** and **parallel** properties, with each step linked to a playbook action.

4 Results and Evaluation

4.1 Illustrative Example

The chosen scenario to explain the application of our approach implementation[7] includes the exploitation of CVE-2021-21277. The remote exploitation of this vulnerability may lead to sensitive information discovery and privilege escalation, allowing the adversary to exploit other vulnerabilities in the system. Figure 3 sketches the network environment. The system includes a vulnerable web server, a terminal with antivirus and firewall software, a SIEM, and an IDS.

Our solution gets the candidate IR actions, matching RE&CT with selected countermeasures. A query retrieves their attributes from the playbook ontology. Considering system software and attacker position, only 14 actions are used in playbook generation. Table 1 lists some of these actions and their parameters.

Playbooks are generated based on the constraints from Section 3.1 for 2 conditions. For the first condition, because of an organization's limited resources, each phase can only be repeated once; this condition results in 28 playbooks. For the second condition, considering that an organization may have more resources available, phases can be repeated multiple times; however, each category can be repeated once per phase. This condition leads to 214 playbooks.

We implement a Pareto front algorithm to select one optimal playbook. The playbooks dominating others and not dominated by any playbook constitute the Pareto front; they are optimal. For the first condition, the algorithm allows us to get 2 optimal playbooks. For the second condition, we get 6 optimal playbooks. In both cases, the solution randomly selects 1 of the optimal playbooks.

Figure 4 represents the optimal playbook selected for the second. In this playbook, 2 identification and 2 containment actions can be executed in parallel; their execution requires different tools. However, the action *Patch vulnerability* has higher priority than the action *Block process by executable format* because it has a higher impact value. Executing actions follows a logical workflow; containment actions follow identification actions. Then, the recovery action *Unblock blocked process* follows. Afterward, the playbook ontology is populated with instances of the selected playbook.

[7] https://github.com/phDimplKS/play

Table 1: List of actions with their parameters' values for CVE-2021-21277

Defensive Technique	Action	Impact	Complexity	Loss	Phase	Category
Process Segment Execution Prevention	find process by executable format	5	3	0.3	Identification	Process
Asset Vulnerability Enumeration	list victims of security alert	4	4	0.2	Identification	General
Process Segment Execution Prevention	list processes executed	4	3	0.2	Identification	Process
Asset Vulnerability Enumeration	put compromised acounts on monitoring	3	7	0.7	Identification	General
Software Update	patch vulnerability	4	3	0.0	Containment	General
Segment Address Offset Randomization	block process by executable format	4	3	0.0	Containment	Process

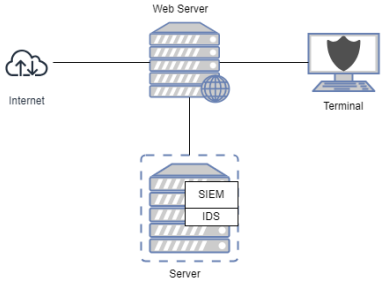

Fig. 3: Example System

Playbook when a phase can be repeated, but a category only repeats once per phase

	Step	React ID	Action	Is automated	Tool
has coa	Step 1	RA2405	Find process by executable format	True	Powershell
parallel	Step 2	RA2001	List victims of security alerts	True	Suricata
has next	Step 3	RA3001	Patch vulnerability	True	WSUS
parallel	Step 4	RA3404	Block process by executable format	True	Group Policy
has next	Step 5	RA5401	Unblock blocked process	True	Group Policy
has next	Step 6	RA6001	Develop incident report	False	NA
has next	Step 7	RA6002	Conduct lessons learned exercise	False	NA

Fig. 4: Optimal Playbook Generated

4.2 Evaluation

We evaluate our approach over 40 CVEs for the 2 different conditions : **First condition** considering that each phase can only be repeated once in a playbook and **Second condition** considering that a phase can be repeated, but a category repeats only once per phase. We base our evaluation on the following metrics:

- The percentage gap between the number of playbooks generated before n and after m applying Pareto optimality using the formula: $(n-m)/n*100$.
- The execution time of our optimal playbook generation process in seconds.

Table 2 compares the average metrics for both conditions. The Pareto front reduces generated playbooks by over 75%. Despite higher generation time in the second condition, model performance remains unaffected mainly due to initially higher playbook numbers.

5 Conclusion

Our proposed approach tackles the SOAR integration problem with a scalable and reusable ontology. It is easier to populate an ontology communicating with a

Table 2: Evaluation of our approach

	No Playbooks before Pareto	**No Playbooks after Pareto**	**First metric: Gap in %**	**Second metric: Time in s**
First condition	26.5	2.1	78.59	7.07
Second condition	8417.5	10.175	82.22	13.01

SOAR than to manage multiple playbook formats. Thanks to all the knowledge bases involved in our approach, the actions of a playbook compose a logical workflow ensuring optimal responses. An optimal playbook is generated each time we launch our system for the same attack scenario on an unchanged system. In the future, we will focus on instantiating the playbook actions on an AG.

References

1. Booth, H., Turner, C.: Vulnerability description ontology (vdo): a framework for characterizing vulnerabilities. Tech. rep., NIST
2. Cichonski, P., Millar, T., Grance, T., Scarfone, K., et al.: Computer security incident handling guide. NIST Special Publication **800**(61), 1–147 (2012)
3. Empl, P., Schlette, D., Stöger, L., Pernul, G.: Generating ics vulnerability playbooks with open standards. International Journal of Information Security pp. 1–16 (2023)
4. Ganin, A.A., Quach, P., Panwar, M., Collier, Z.A., Keisler, J.M., Marchese, D., Linkov, I.: Multicriteria decision framework for cybersecurity risk assessment and management. Risk Analysis **40**(1), 183–199 (2020)
5. Hutschenreuter, H., Çakmakçi, S.D., Maeder, C., Kemmerich, T.: Ontology-based cybersecurity and resilience framework. In: ICISSP. pp. 458–466 (2021)
6. Islam, C., Babar, M.A., Nepal, S.: Automated interpretation and integration of security tools using semantic knowledge. In: Advanced Information Systems Engineering: 31st International Conference, CAiSE 2019, Rome, Italy, June 3–7, 2019, Proceedings 31. pp. 513–528. Springer (2019)
7. Jordan, B., Thomson, A.: Cacao security playbooks version 1.0. OASIS, Committee Specification **2** (2021)
8. Komal, S., Zakeya, N., Raphael, R., Harit, A., Mohammadreza, R., Marin, L., Larisa, S., Ian, W.: Adarma auto-detection and auto-remediation of microservice anomalies by leveraging large language models. In: Proceedings of the 33rd Annual International Conference on Computer Science and Software Engineering. pp. 200–205 (2023)
9. Mern, J., Hatch, K., Silva, R., Hickert, C., Sookoor, T., Kochenderfer, M.J.: Autonomous attack mitigation for industrial control systems. In: 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). pp. 28–36. IEEE (2022)
10. Moreira, G.B., Calegario, V.M., Duarte, J.C., dos Santos, A.F.P.: Csiho: An ontology for computer security incident handling. In: Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais. pp. 1–14. SBC (2018)
11. Syed, Z., Padia, A., Finin, T., Mathews, L., Joshi, A.: Uco: A unified cybersecurity ontology. UMBC Student Collection (2016)
12. Van Veldhuizen, D.A., Lamont, G.B., et al.: Evolutionary computation and convergence to a pareto front. In: Late breaking papers at the genetic programming 1998 conference. pp. 221–228. Citeseer (1998)