Undefined 1 (2014) 1–5
IOS Press

Attack Graphs & Subset Sabotage Games

Davide Catta [a,*] Jean Leneutre [a] and Vadim Malvone [a]
[a] *LTCI, Télécom Paris, Institut Polytechnique de Paris, Paris, France*
E-mail: {davide.catta, jean.leneutre, vadim.malvone}@telecom-paris.fr

Abstract. We consider an extended version of sabotage games played over Attack Graphs. Such games are two-player zero-sum reachability games between an Attacker and a Defender. This latter player can erase particular subsets of edges of the Attack Graph. To reason about such games we introduce a variant of Sabotage Modal Logic (that we call Subset Sabotage Modal Logic) in which one modality quantifies over non-empty subset of edges. We show that we can characterize the existence of winning Attacker strategies by formulas of Subset Sabotage Modal Logic.

Keywords: Attack Graphs, Sabotage Games, Logics in Games

1. Introduction

Modern systems are inherently complex and security plays a crucial role. The main challenge in developing a secure system is to come up with tools able to detect vulnerabilities and unexpected behaviors at a very early stage of its life cycle. These methodologies should be able to measure the grade of resilience to external attacks. Crucially, the cost of repairing a system flaw during maintenance is at least two orders of magnitude higher, compared to fixing at an early design stage [9].

In the past fifty years, several solutions have been proposed to check system reliability and, in particular, system security. In this area a story of success is the use of *formal methods* techniques [9]. They allow checking whether a system is correct by formally checking whether a mathematical model of it meets a formal representation of its desired behavior. Recently, classic approaches such as *model checking* and automata-theoretic techniques, originally developed for monolithic systems [8,20], have been meaningfully extended to handle *open* and *multi-agent systems* [1,17,21,24,25]. These are systems that encapsulate the behavior of two or more rational agents interacting among them in a cooperative or adversarial way, aiming at a designed goal [18].

In system security checking, a malicious attack can be seen as an attempt of an Attacker to gain an unauthorized resource access or compromise the system's integrity. In this setting, *Attack Graph* [22] is one of the most prominent attack models developed and receiving much attention in recent years. This encompasses a graph where each state represents an Attacker at a specified network location and edges represent state transitions, i.e., attack actions by the Attacker. Then, it is the system's duty to prevent unauthorized access from the Attacker in each state of the graph. Said more precisely, the Attacker's goal is to reach a certain state of the Attack Graph by traveling through its edges, while the Defender's goal is to prevent him from doing so. To do this, the Defender can dynamically deploy countermeasures preventing the attack to succeeds. The attacks are represented by the edges of the Attack Graph. If the Defender deploys a countermeasure and such countermeasure is successful, the Attacker will no longer be able to use an attack. We can formalize such a scenario as a two-player turn based game between the Defender and the Attacker, where in turn the latter moves along adjacent states (w.r.t. the Attack Graph under exam) and the former inhibits some attacks by erasing some subset of edges of the Attack Graph itself. The goal of the Defender is to block the Attacker to reach some designated states, while not blocking the entire system functionality. The kind of scenario that we have just sketched is an exam-

*Corresponding author.

ple of **extended** *sabotage game* [31]. Sabotage games were introduced by van Benthem in 2005 with the aim of studying the computational complexity of a special class of graph-reachability problems. Namely, graph reachability problems in which the set of edges of the graph became thinner and thinner as long as a path of the graph is constructed. To reason about sabotage games, van Benthem introduced Sabotage Modal Logic. Such a logic is obtained by adding to the $\Diamond$-modality of classical modal logic another modality $\blacklozenge$. Let G be a directed graph and s one of its vertex (or states); the intended meaning of a formula $\blacklozenge\varphi$ is " $\blacklozenge\varphi$ is true at a state s of G iff φ is true at s in the graph obtained by G by erasing an edge e".

Our sabotage games differ from the one introduced by van Benthem because one of the players can erase *an entire subset of edges of a given graph*. To reason about such games, we introduce a variant of Sabotage Modal Logic that we call, for lack of wit, Subset Sabotage Modal Logic (SSML for short). The logic SSML is obtained by adding a modality $\blacklozenge^{\subset}$ to the language of classical modal logic. The intended meaning of a formula $\blacklozenge^{\subset}\varphi$ is " $\blacklozenge^{\subset}\varphi$ is true at a state s of a directed graph G iff φ is true at s in the graph G' that is obtained from G by erasing a non-empty set of its edges". We prove that the model-checking problem for SSML is decidable and that the existence of an Attacker winning strategy over an Attack Graph can be expressed by using an SSML formula.

Structure of the work. The rest of the paper is structured as follows. In Sec 2 we discuss the related works. In Sec 3 we briefly present Sabotage Modal Logic. In Sec. 4 we present Attack Graphs, propose a formal definition of such objects in terms of Kripke structures and introduce, informally, subset sabotage games on Attack Graphs. Then, in Sec 5, we formally define the class of subset sabotage games on Attack Graphs that we are interested on. We define plays on such games and Attacker winning strategies. In the penultimate section (Sec 6), we introduce Subset Sabotage Modal Logic and show that the model checking problem for such logic is decidable. To prove such a result, we reduce the model checking problem of SSML to the one of SML by giving a translation of SSML-formulas into SML formulas. We then show that the existence of an Attacker winning strategies for a subset sabotage game over an Attack Graph, is equivalent to the satisfiability of a certain SSML formula at the root of the considered Attack Graph. The last section concludes by discussing possible future works.

2. Related Work

Several existing works have proposed different game-theoretic solutions for finding an optimal defense policy based on Attack Graphs. Most of these approaches do not use formal verification to analyze the game, but rather try to solve them using analytic and optimization techniques. The work in [11,12] studies the problem of hardening the security of a network by deploying honeypots to the network to deceive the Attacker. They model the problem as a Stackelberg security game, in which the attack scenario is represented using Attack Graphs. The authors in [26] tackle the problem of allocating limited security countermeasures to harden security based on attack scenarios modeled by Bayesian Attack Graphs using partially observable stochastic games. They provide heuristic strategies for players and employ a simulation-based methodology to evaluate them. The work in [32] proposes an approach to select an optimal corrective security portfolio given a probabilistic Attack Graph. They define a Bayesian Stackelberg game that they solve by converting it into Mixed-Integer Conic Programming (MICP) optimization problem.

Since the games we introduce are a variant of sabotage games, our work is indebted to those dedicated to this type of games and to Sabotage Modal Logic [3, 23,31]. In particular, the authors of [23] shows the decidability of the model-checking problem for Sabotage Modal Logic. Such a result is important to our work because we use it to show the decidability of the model-checking problem for Subset Sabotage Modal Logic. In [3] the authors develop a complete proof system for Sabotage Modal Logic, they define and study the notion of bisimulation, and they present an extensive discussion of sabotage games and their characterization via the logic.

The present work is an extended version of the material already published in [7]

3. Sabotage Modal Logic

In this section, we briefly present Sabotage Modal Logic (SML, for short). Such logic was proposed in 2005 by Van Benthem [31] as a format for analyzing games that modify the graphs they are played on. SML was later investigated in a series of papers. Before entering into the matter of SML, let us fix some notation and terminology that will be used in the following.

Notation & Terminology. *Given a sequence ρ, we denote its length as $|\rho|$, and its $(j+1)$-th element as ρ_j. For $j \leq |\rho|$, let $\rho_{\geq j}$ be the suffix of ρ starting at ρ_j and $\rho_{\leq j}$ the prefix $\rho_0 \cdots \rho_j$ of ρ. The empty sequence will be denoted by ϵ. A* tree *is a (finite or infinite) connected directed graph, with one node designated as the root, and in which every non-root node as a unique parent (s is the parent of t, and t is the child of s is there is an edge from s to t) and the root has no parent. The arity of a node x in a three $\mathcal{T}$ is the number of child of x. A path $\mathcal{P} = x_0, x_1, \ldots$ is a (finite or infinite) sequence of nodes such x_i is the parent of x_{i+1} for all $i > 0$. A branch is a path that is maximal and whose first node is the root . A node x is an ancestor of a node y if there is a path containing x whose last element is y (remark that every node is the ancestor of itself).*

Given a non-empty set $\mathcal{P}$ of atomic formulas, and a finite non-empty set Σ of labels, we define SML formulas by the following grammar:

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond_a \varphi \mid \blacklozenge_a \varphi$$

where $p \in \mathcal{P}$ and $a \in \Sigma$. We define $\top \doteq \neg\bot$. If φ and ψ are formulas, we define $\varphi \rightarrow \psi \doteq \neg\varphi \vee \psi$, $\phi \wedge \psi \doteq \neg(\neg\varphi \vee \neg\psi)$, $\Box_a \psi \doteq \neg \Diamond_a \neg\varphi$ and $\blacksquare_a \doteq \neg \blacklozenge_a \neg\varphi$. The size $|\varphi|$ of a formula φ is inductively defined. It is 1 if φ is an atomic formula or $\bot$, it is $|\psi| + 1$ if $\varphi = \circ\psi$ with $\circ \in \{\neg, \Diamond_a, \blacklozenge_a\}$ and $max\{|\varphi_1|, |\varphi_2|\} + 1$ if $\varphi = \varphi_1 \vee \varphi_2$.

We now define the structures that will serve as interpretation of SML-formulas

Definition 1. *A rooted Kripke structure is a tuple $M = \langle S, s_0, \Sigma, R, V\rangle$ where S is a non-empty set of states, $s_0 \in S$ is the initial state, Σ is a finite set of labels, $R \subseteq S \times \Sigma \times S$ is the transition relation, and $V : S \rightarrow 2^{\mathcal{P}}$ is the evaluation function, assigning a set of atomic formulas to any state $s \in S$.*

If $M = \langle S, s_0, \Sigma, R, V\rangle$ is a Kripke structure and $E \subseteq R$, we write $M \setminus E$ to denote the Kripke structure obtained by erasing the subset E from the relation R. If $e = (s, a, s') \in R$ we say that e is a *labeled edge* or simply an edge, and that a is the label of e. We denote by R_a the subset of labeled edges of R whose label is a, that is $R_a = \{e \in R \mid e \in S \times \{a\} \times S\}$.

The notion of satisfaction of a formula φ at a given state s of a Kripke structure M (written $M, s \models \varphi$) is inductively defined as follows:

$M, s \models \bot$ never;
$M, s \models p$ iff $p \in V(s)$;
$M, s \models \neg\varphi$ iff not $M, s \models \varphi$ (notation $M, s \not\models \varphi$);
$M, s \models \varphi \vee \psi$ iff $M, s \models \varphi$ or $M, s \models \psi$;
$M, s \models \Diamond_a \varphi$ iff there is a $s' \in S$ such that $(s, a, s') \in R$ and $M, s' \models \varphi$;
$M, s \models \blacklozenge_a \varphi$ iff there is $(s_1, a, s_2) \in R$ such that $M \setminus \{(s_1, a, s_2)\}, s \models \varphi$.

We say that a formula φ is true in a rooted Kripke structure M (written $M \models \varphi$) iff φ is true at the initial state of M. As we can see from the above definition, the meaning of the boolean connectives of SML logic is the standard one. Equally, the meaning of the $\Diamond_a$ -connective is the standard meaning in modal logic: a formula $\Diamond_a \varphi$ is true at a given state s of a Kripke structure whenever s is adjacent (with respect to an edge labeled by a) to a vertex s' for which the property expressed by φ is true. The meaning of the $\blacklozenge_a$ -connective can be spelled out as follows: a formula $\blacklozenge_a \varphi$ is true at a given state s of a Kripke structure M whenever φ is true at s in the Kripke structure M' that is obtained by erasing an edge (s', a, s'') from R in M.

Definition 2. *The model checking problem for Sabotage Modal Logic consists of the following data and question:*

Data 1*: an SML formula φ,*
Data 2*: a rooted finite Kripke structure M,*
Question*: Is it the case that $M \models \varphi$?*

The proof of the following theorem can be found in [23].

Theorem 1. *The model checking problem for Sabotage Modal Logic is PSPACE-complete.*

4. Attack Graphs

The term Attack Graph has been first introduced by Phillips and Swiler [29]. Attack Graphs represent the possible sequence of attacks in a system as a graph. An Attack Graph may be generated by using the following pieces of information:

1. a description of the system architecture (topology, configurations of components, etc.);
2. the list of the known vulnerabilities of the system;
3. the Attacker's profile (his capabilities, password knowledge, privileges, etc.) and attack templates (Attacker's atomic action, including preconditions and postconditions).

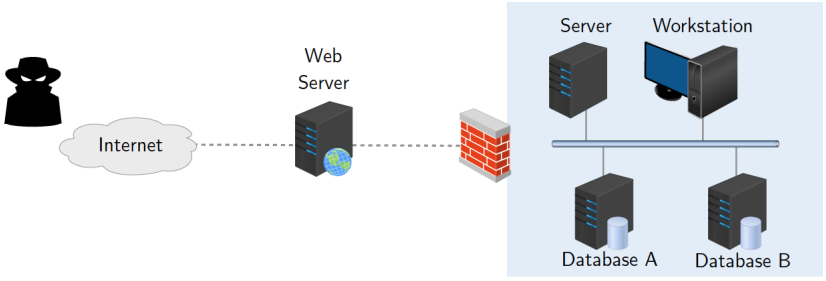


Fig. 1. An illustrating LAN architecture example.

An attack path in the graph corresponds to a sequence of atomic attacks. Several works have developed this approach, see e.g., [16,27,28,30**?**], and [19] for a survey. Each of the previously cited works introduced its own Attack Graph model with its specificity, and thus there is no standard definition of an Attack Graph. However, all introduced models can be mapped into a *canonical Attack Graph* as introduced in [15]. It is a labeled, oriented graph where:

1. each node represents both the state of the system (including existing vulnerabilities) and the state of the Attacker including constants (Attacker skills, financial resources, etc.) and variables (knowledge of the network topology, privilege level, obtained credentials, etc.);
2. each edge represents an action of the Attacker (a scan of the network, the execution of an exploit based on a given vulnerability, access to a device, etc.) that changes the state of the network or the states of the Attacker; an edge is labelled with the name of the action (several edges of the Attack Graph may have the same label).

An Attack Graph is said *complete* whenever the following condition holds: for every state q and for every atomic attack *att*, if the preconditions of the atomic attack hold in q, then there is an out coming edge from q labeled with *att*.

By abstracting all the data of the above discussion, one can see an Attack Graph as a directed graph together with a labeling of its vertices and edges. The labeling of vertices is used to specify which properties (the kind of properties mentioned in 1) are true at a certain vertex, while the edge labeling specifies the name of the action of the Attacker. As we have seen in the example above, it is also useful to specify the set of Attacker's target states. We thus define an Attack Graph as follows:

Definition 3. *Suppose that the set $\mathcal{P}$ contains an atomic proposition* win*. An **Attack Graph** is a tuple $AG = \langle S, s_0, \Sigma, R, V, T\rangle$ where:*

- *$\langle S, s_0, \Sigma, R, V\rangle$ is a rooted Kripke structure where the set S of states is finite and for all $a \in \Sigma$ if $(s, a, s') \in R$ then there is no $b \in \Sigma$ such that $(s, b, s') \in R$.*
- *T is a non-empty subset of S such that* win $\in V(s)$ *for all $s \in T$.*

The set T represent the set of target states of the Attacker.

Example 1. *Consider the following scenario: an enterprise has a local area network (LAN) that features a* Server*, a* Workstation *and two databases A and B. The LAN also provided a* Web Server*. Internet access to the LAN is controlled by a firewall. Such a scenario is depicted in Figure 1. Suppose that we know some vulnerabilities and that we have established that a malevolent user can make the attack listed in Table 1, e.g., by making att_2 an Attacker can exploit a vulnerability related to the* Server*: as a precondition, the Attacker needs to have root access to the* Web Server *and, as a postcondition, he will obtain root access to the* Server*.*

Then we can construct an Attack Graph built from this set of atomic attacks and collecting possible attack paths as depicted in Figure 2[1]*. The Attacker's initial state is a node in the Attack Graph. Let us suppose that the Attacker is in state s_1 and wants to reach state s_4. To get to this target, she can perform the sequences of atomic attacks att_2, att_4 or att_3, att_2, att_4.*

[1]This Attack Graph is not complete w.r.t. our previous description, since some possible sequences of atomic attacks are not listed: for instance $att_1, att_2, att_3, att_5$ are not taken into account.

Attack	Location	Precondition	Postcondition	Counter measure
att_1	Web Server		$web\_server : root$	_
att_2	Server	$web\_server : root$	$server : root$	c_2
att_3	Workstation	$web\_server : root$	$password : 1234$	_
att_4	Database A	$server : root$	$databaseA : root$	c_4
att_5	Database B	$server : root \wedge$ $password : 1234$	$databaseB : root$	c_5

Table 1
Atomic attacks and countermeasures over the LAN of Figure 1.

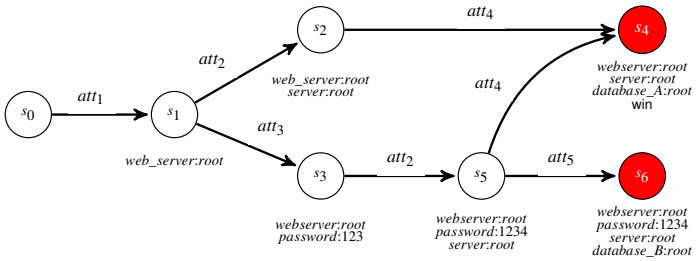


Fig. 2. Example of Attack Graph, the atomic proposition satisfied at a given state are listed below the state itself.

5. Attack Graphs & Games

In the previous section, we saw that given a specific description of a system together with its vulnerabilities, we can generate a graph representing the dynamics of attacks that are possible over the system. Given a set of target states over such a graph, one can ask whether there is a path from an initial state to one of these target states, i.e., by reasoning over Attack Graphs, we can encode a security problem as a graph-reachability problem. Let us make one step more by adding a dynamic to such reachability problems. An Attack Graph represents a sequence of possible actions made by an Attacker to reach a specific goal. Let us add another character to this story, the Defender, whose objective is to counter the attack. Suppose that she has the power to dynamically deploy a predefined set of countermeasures: for instance by reconfiguring the firewall filtering rules, or patching some vulnerabilities, that is by removing one or several preconditions of an atomic attack. A given countermeasure c will prevent the Attacker from longing a given attack att: deploying c is equivalent to removing all the edges in the Attack Graph labeled with att. In real situations, due to budget limitations or technical constraints, the set of available countermeasures may not cover all atomic attacks. Now that we have specified what is the Defender's power, we can consider a turn-based game between an Attacker and a Defender. Both players play on an Attack Graph $\langle S, s_0, \Sigma, R, V, T \rangle$. The Attacker's goal is to reach one of the states in T, while the Defender's goal is to prevent him from doing so. The Defender starts the game by selecting a certain countermeasure. By choosing such a countermeasure, she deletes a subset of the edges of the Attack Graph. The Attacker takes his turn and moves from s_0 to one of its successors along the edges that have not been erased (if any). The game evolves following this pattern. The Attacker won if he can reach one of the states in T in a finite number of moves, the Defender wins otherwise.

Example 2. *Consider again the Attack Graph of Figure 2. Suppose that the initial state is s_1. Suppose that the Defender has at her disposal a countermeasure c_2 for attack att_2, c_4 for attack att_4, and c_5 for attack att_5, but no one for the attacks att_1 and att_3 as reported in the last column of Table 1. The Defender starts the game by deploying countermeasure c_3. The only edge of the Attack Graph labeled by att_3 is the one going from s_1 to s_3; consequently, such edge is erased from the Attack Graph and the Attacker can only move to s_2.*

The Defender takes again the turn and deploys countermeasure c_2. There are two edges that are labeled by att_2 and both are erased from the Attack Graph. The Attacker moves from s_2 to s_4 and, since s_4 is a target state, she wins the game.

There is a natural way to look at the games described above: at each round, the Attacker can follow an edge from his current position in the given graph, but the Defender can choose a new graph (which is a subgraph of the current graph) missing a subset of edges. To precisely define such plays, we first need an auxiliary notion. We know that it is reasonable to assume that the Defender is unable to counter each of the Attacker's possible attacks. At each step of the game, she can only deactivate a relation contained in a certain subset of the Attack Graph relation set. As mentioned above, we can consider that the Defender selects a sub-graph of the current Attack Graph whenever it is her turn to move. According to the limitation set out above, this sub-graph must be obtained by deleting some specific subset of the set of labeled edges. We define this notion as follows.

Definition 4. *Let $AG = \langle S, s_0, \Sigma, R, V, T\rangle$ be an Attack-graph and $\Delta \subseteq \Sigma$. The Attack Graph AG' is Δ-reachable from AG iff $AG' = \langle S, s_0, \Sigma, R \setminus R_b, V, T\rangle$ for some $b \in \Delta$ or $AG' = AG$.*

We have now all the ingredients to define a play in our game.

Definition 5. *Let $AG = \langle S, s_0, \Sigma, R, V, T\rangle$ be an Attack-graph and Δ a non-empty subset of Σ. A Δ-play ρ is a non-empty sequence ρ of length at least 2 where $\rho_0 = AG$, $\rho_1 = s_0$ and for all $2 < j \leq |\rho|$:*

1. *ρ_j is an Attack Graph if j is even.*
2. *ρ_j is a state of S if j is odd.*
3. *ρ_j is Δ-reachable from ρ_{j-2}, if j is even.*
4. *$(\rho_{j-2}, a, \rho_j) \in R$ is an edge in the Attack Graph $AG' = \rho_{j-1}$ for some $a \in \Sigma$, if j is odd.*

Let ρ be a finite Δ-play whose last element is state s. We say that the Attack Graph AG' is legal for ρ iff win $\notin V(s)$ *and the sequence obtained by concatenating ρ to AG' is a Δ-play.*

Remark that given any Δ-play ρ and any even natural number $j < |\rho|$, the sequence $\rho_{\geq j}$ is a Δ-play over the Attack Graph obtained by setting the state ρ_{j+1} as initial state of the Attack Graph ρ_j. If ρ is Δ-play we denote by ρ^A (resp. ρ^S) the subsequence of ρ in which only Attack Graphs (resp. states) appears.

Definition 6. *Let ρ be a Δ-play over an Attack Graph AG. We say that ρ is won by the Attacker iff $|\rho| = 2n$ for some $n \in \mathbb{N}$ and there is no Attack Graph AG' legal for ρ.*

Said plainly: the Attacker wins the game whenever he reaches one of his target states.

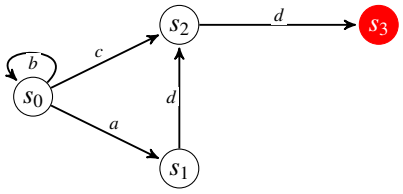


Fig. 3. Another example of Attack Graph. The red state is the only state in T.

According to the above definition of play, the same state can appear many times in the same play. For instance, consider the Attack Graph AG shown in Figure 3. Let AG' be the Attack Graph obtained by deleting from AG the edge labeled by c. Consider the following $\{c\}$-play ρ over AG:

$$\begin{array}{cccccccccccc} AG & s_0 & AG' & s_0 & AG' & s_0 & AG' & s_1 & AG' & s_2 & AG' & s_3 \\ \rho_0 & \rho_1 & \rho_2 & \rho_3 & \rho_4 & \rho_5 & \rho_6 & \rho_7 & \rho_8 & \rho_9 & \rho_{10} & \rho_{11} \end{array}$$

We can see that the vertex s_0 appears three times in such a play. We define another class of plays in which this phenomenon cannot happen.

Definition 7. *Let ρ be a Δ-play over an Attack Graph AG. We say that ρ is* stubborn *whenever for any $0 \leq j \leq |\rho|$ if j is odd then there is no $i < j$ such that $\rho_i = \rho_j$.*

The definition above precludes the Attacker from choosing the same vertex of an Attack Graph in two different turns of the play. As an example, the $\{c\}$-play presented above is not a $\{c\}$ stubborn play over AG because $\rho_1 = \rho_3 = \rho_5$.
Every stubborn Δ-play is a Δ-play. If ρ is a stubborn Δ-play over an Attack Graph AG, then $|\rho| \leq 2(|S| - 1)$, where S is the set of states of AG. This is because if s and s' are two distinct states of AG and there is a path $p_0 \cdots p_k$ where all the states p_i are distinct, then $k \leq |S| - 1$.

A strategy is usually defined as a function. A function that specifies, at each moment of the game, which move a player must play according to the moves pre-

viously played (the history of the game). A strategy is *winning* when the player who is following the strategy wins, whatever the history of the game is. We choose another equivalent definition of strategy. We informally describe how a strategy should operate and then formalize this notion. Imagine being engaged in a game $\mathcal{G}$, that the last move of $\mathcal{G}$ was played according to the strategy, and that it is now the Opponent's turn to play. The Opponent could extend the game in different ways: for example, if you are playing chess, you are white, and you just made your first move by moving a pawn to a certain position of the chessboard, black can in turn move a pawn or move a horse. If you are playing according to the strategy, the strategy should tell you how to react against either type of move. Therefore, a strategy can be viewed as a tree in which each node is move of the game, your Opponent's move have at most one daughter and your moves have as many daughters as there are available moves (with respect to the considered play) for the Opponent. We therefore define strategies as follows:

Definition 8. *An Attacker Δ winning strategy $\mathcal{S}$ for an Attack Graph AG is a finite tree in which each branch is a Δ-play won by the Attacker and moreover:*

1. *for all node v of $\mathcal{S}$: if v is an Attack Graph then v has a unique child;*
2. *for all node v of $\mathcal{S}$: if v is a state of AG then v has as many children as there are Δ-reacheable Attack Graphs from the parent of v.*

A strategy is stubborn iff every branch of the strategy is a stubborn play.

Remark that any winning Stubborn strategy is a winning strategy by definition.

Let $\alpha = AG_0 \cdots AG_n$ be a finite non empty sequence of Attack Graphs. Suppose that $AG_0 = \langle S, s_0, \Sigma, R, T\rangle$ and for all $1 \leq i \leq n$, $AG_i = \langle S, s_e, \Sigma, (R \setminus E_1 \cup \cdots \cup E_i), T\rangle$ where for each i, E_i is an eventually empty subset of R. If E is a set of edges such that for all $e \in E$ e is of the form (s, a, s') for some $s, s' \in S$ and $a \in \Sigma$, then we denote by α_{+E} the sequence $AG'_1 \cdots AG'_n$ where $AG'_0 = \langle S, s_0, \Sigma, R \cup E, T\rangle$ and for all $1 \leq i \leq n$, $AG_i = \langle S, s_0, \Sigma, ((R \cup E) \setminus E_1 \cup \cdots \cup E_i), T\rangle$.

Let $\mathcal{S}$ be a winning Attacker Δ-strategy for some Attack graph AG. A **critical section** of $\mathcal{S}$ is a sequence of nodes $s' \cdots s'$ that starts and ends with a node $s' \in S$ and that lies along one of the branches of $\mathcal{S}$. Clearly $\mathcal{S}$ is stubborn iff $\mathcal{S}$ does not contain any critical section.

Proposition 1. *For any Attack Graph AG if there is a winning Attacker Δ-strategy $\mathcal{S}$ for AG then there is a winning Attacker stubborn Δ-strategy $\mathcal{S}'$ for AG.*

The proof is by induction on the number n of critical sections of $\mathcal{S}$. If $n = 0$, then $\mathcal{S}$ is already a stubborn strategy, and we are done. Suppose that the hypothesis of the proposition holds for any winning strategy having at most n critical sections, and let $\mathcal{S}$ be a strategy with $n + 1$ critical sections. Choose a critical section $s' \cdots s'$ of $\mathcal{S}$. In $\mathcal{S}$ there is a finite number of branches $\rho^1, \ldots, \rho^n$ for $n \geq 1$ in which such a critical section is included, that is: each of the ρ^j has the form $\rho' \; s' \cdots s' \, \tau$ for some alternated sequence of Attack Graphs and states τ, and some path ρ' of $\mathcal{S}$. The idea of the proof is to cut the critical section $s' \cdots s'$ from any of these branches and modify the Attack Graphs of the obtained sequence to obtain a play. Such a modification is obtained by using the construction define below Definition 8 . Let us denote by $\mathbf{c}$ the critical section $s' \cdots s'$ and let $R_{\mathbf{c}}$ be the set of edges deleted by the Defender along $\mathbf{c}$. For any $\rho^1 \ldots \rho^n$ we define $\rho^j_\star = \rho' \; s' \tau_\star$ where $\tau^A_\star = \tau^A_{+R_{\mathbf{c}}}$ and $\tau^S_\star = \tau^S$. Any $\rho^j_\star$ is a Δ-play won by the Attacker. Let $\mathcal{S}_\star$ be the tree of plays induced by the transformation above. Clearly, $\mathcal{S}_\star$ is a winning Attacker strategy for AG that contains less critical sections than $\mathcal{S}$. We apply the induction hypothesis and we obtain that there is a winning stubborn strategy $\mathcal{S}'$ for AG.

6. Games & Subset Sabotage Modal Logic

In this section, we explain why we cannot directly use SML to reason about the above introduced games. We then introduce the Subset Sabotage Modal Logic (SSML) precisely to rectify this problem and we study its properties. In particular, we prove that the model checking problem for SSML is decidable and that the existence of an Attacker Winning strategy is equivalent to the satisfiability of a certain SSML formula.

6.1. What is the problem with SML?

Plays, that we defined in the previous sections, are nothing more than runs in an insidious sabotage game. In a sabotage game, one of the two players can delete an edge of the graph when it is her turn to move. In our games, one of the two players can delete a *subset* of edges of the graph when it is her turn to move. In [3] the authors claims that, by using our terminol-

ogy, the existence of an Attacker winning strategy for a Sabotage Game over a finite rooted Kripke structure $M = \langle S, s_0, \Sigma, R, V\rangle$ can be expressed by using a particular SML-formula. Such formula is defined by induction on n,

$$\lambda_n = \begin{cases} \mathsf{win} & \text{if } n = 0 \\ \Diamond \top \wedge \blacksquare\Diamond(\lambda_{n-1} \vee \mathsf{win}) & \text{otherwise} \end{cases} \quad (1)$$

where $\blacksquare\varphi \doteq \bigwedge_{a\in\Sigma} \blacksquare_a \varphi$ and $\Diamond\varphi \doteq \bigvee_{a\in\Sigma} \Diamond_a \varphi$ for a given SML formula φ, and $\Diamond\top$ is used to check that the $\blacksquare$ is not satisfied because some relation is empty.

More precisely, if $M = \langle S, s_0, \Sigma, R, V\rangle$ is a Kripke structure such that $\mathsf{win} \in V(s)$ for some $s \in S$ and $|S| = n \geq 1$ then $M, s_0 \models \mathsf{win} \vee \lambda_{n-1}$ if and only if there is an Attacker winning strategy for the sabotage game played over M. Such a result is false in the case of the particular class of sabotage games that we have defined in the previous section, precisely because the Defender erases a subset of R at each step of the game.

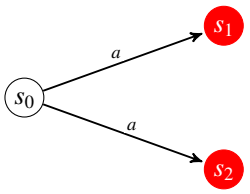


Fig. 4. An additional example of Attack Graph. The red states are the only ones in T.

For instance, consider the Attack Graph depicted in Figure 4. We can see that there is no $\{a\}$-winning strategy over M. If the Defender erases any edge labeled with a, the Attacker would not be able to move from s_0 to one of the winning states s_1 or s_2. On the contrary, the formula $\mathsf{win} \vee \lambda_2 = \mathsf{win} \vee (\Diamond \top \wedge \blacksquare\Diamond(\lambda_1 \vee \mathsf{win}))$ is true at s_0 in M. This is because if the top-most a-edge is removed, the Attacker can pass from the lowermost edge to reach a winning state, and he can pass from the top-most one if the lower one is removed.

6.2. Subset Sabotage Modal Logic

To speak about our particular games, we introduce a variant of Sabotage Modal Logic. We call such variant of Sabotage Modal Logic, Subset Sabotage Modal Logic.

Definition 9. *Given a non-empty set $\mathcal{P}$ of atomic propositions and a finite non-empty set Σ of labels, formulas of Subset Sabotage Modal Logic (SSML, for short) are defined by the following grammar:*

$$\varphi ::= p \mid \bot \mid \neg\varphi \mid \varphi \vee \varphi \mid \Diamond_a \varphi \mid \blacklozenge^{\subseteq}_a \varphi$$

where $p \in \mathcal{P}$ and $a \in \Sigma$. Given a rooted Kripke structure $M = \langle S, s_0, \Sigma, R, V\rangle$, recall that for $a \in \Sigma$, R_a denote the subset of R defined by $\{e \in R \mid e \in S \times \{a\} \times S\}$. The definition of satisfaction of an SSML formula at a state s of a Kripke structure M is defined inductively. Such definition is the same as SML for atomic proposition, negation, disjunction and the diamond modality. It is defined as follows for the $\blacklozenge^{\subseteq}_a$-modality:

> *$M, s \models \blacklozenge^{\subseteq}_a \varphi$ iff there is a non-empty subset E of R_a such that $M \setminus E, s \models \varphi$*

where $M \setminus E$ denotes the structure M from which we have erased the subset E of edges. If φ is an SSML formula we define $\blacksquare^{\subseteq}_a \varphi \doteq \neg \blacklozenge^{\subseteq}_a \neg\varphi$.

Remark 1. *$M, s \models \blacksquare^{\subseteq}_a \varphi$ if and only if for any non-empty subset E of R_a we have that $M \setminus E, s \models \varphi$, R_a included. This implies that if $M, s \models \blacksquare^{\subseteq}_a \varphi$ then $M \setminus R_a, s \models \varphi$.*

We now define the model checking problem for SSML.

Definition 10. *The model checking problem for SSML consists of the following data and question:*

> ***Data 1**: a finite rooted Kripke structure M,*
> ***Data 2**: a SSML formula φ,*
> ***Question**: is it the case that $M \models \varphi$?*

In what follows, we show that the model checking problem for SSML is decidable. To do so, we reduce the model-checking problem for SSML to the model-checking problem for SML. The idea of the proof is the following: we are considering a finite Kripke structure. As a consequence, any of its non-empty subset of edges is finite. We give a name n_e to each edge e of M obtaining a Kripke structure M'. We remark that $M \models \blacklozenge^{\subseteq}_a \varphi$ iff there is a finite, non-empty subset of edges $\{e_1, \dots e_n\}$ of R_a such that $M \setminus \{e_1, \dots, e_n\} \models \varphi$. If φ is a SML formula then $M' \models \blacklozenge_{n_{e_1}} \cdots \blacklozenge_{n_{e_n}} \varphi$, that is $M' \setminus \{n_{e_1}, \dots n_{e_n}\} \models \varphi$, where each n_{e_i} is the name of edge e_i. Thus, we need to give a translation from SSML-formulas to SML formulas. Such translation will be parametrized by Kripke structures because we need to consider their subsets of edges.

Definition 11. *Let $M = \langle S, s_0, \Sigma, R, V\rangle$ be a finite rooted Kripke structure. For all $a \in \Sigma$, let $f : R \to \mathbb{N}$ be an injective function associating to each member e of R a natural number n_e. We define the structure $M^{\subset} = \langle S^{\subset}, s_0^{\subset}, \Sigma^{\subset}, R^{\subset}, V^{\subset}\rangle$ as follows:*

- *the set of states of $M^{\subset}$ and its initial state are the same of M;*
- *the set of labels $\Sigma^{\subset}$ is $\{n_e \mid e \in R\}$;*
- *$(s, n_e, s') \in R^{\subset}$ iff for some $a \in \Sigma$ $f(s, a, s') = n_e$;*
- *the evaluation function $V^{\subset}$ is V.*

By the definition above, if $n_e \in \Sigma^{\subset}$, there is exactly one edge $e \in R_{n_e} = \{e \in R^{\Sigma} \mid e \in S \times \{n_e\} \times S\}$.

Remark 2. *Let M be a finite rooted Kripke Structure, E any non-empty subset of R and $M^{\subset}$ the corresponding Kripke structure introduced above. For any non-empty subset $E = \{e_1, \ldots, e_n\}$ of R: $(M \setminus \{e_1, \ldots, e_n\})^{\subset} = M^{\subset} \setminus \{n_{e_1}, \ldots, n_{e_n}\}$.*

If $E = \{e_1, \ldots, e_n\}$ is a subset of edges of M and φ is a formula, we write $\blacklozenge_{n_E}\varphi$ as a short-cut for the formula $\blacklozenge_{n_{e_1}} \cdots \blacklozenge_{n_{e_n}}\varphi$ where each n_{e_i} is the label corresponding to e_i in $M^{\subset}$.

We define a function that maps an SSML formula φ to an SML formula $(\varphi)^{\star}_{M}$. Such a function takes as arguments a Kripke Structure M and an SSML formula φ, and gives as result an SML formula $(\varphi)^{\star}_{M}$. The function is defined on the structure of φ, as follows:

$$
\begin{aligned}
(p)^{\star}_{M} &= p \\
(\bot)^{\star}_{M} &= \bot \\
(\neg\varphi)^{\star}_{M} &= \neg(\varphi)^{\star}_{M} \\
(\varphi \vee \psi)^{\star}_{M} &= (\varphi)^{\star}_{M} \vee (\psi)^{\star}_{M} \\
(\Diamond_a \varphi)^{\star}_{M} &= \bigvee_{e \in R_a} \Diamond_{n_e} (\varphi)^{\star}_{M} \\
(\blacklozenge^{\subset}_a \varphi)^{\star}_{M} &= \bigvee_{E \in 2^{R_a} \setminus \emptyset} (\blacklozenge_{n_E} (\varphi)^{\star}_{M})
\end{aligned}
$$

Lemma 1. *For any $M = \langle S, s_0, \Sigma, R, V\rangle$, for any SSML formula φ, for any state $s \in S$: $M, s \models \varphi$ iff $M^{\subset}, s \models (\varphi)^{\star}_{M}$*

The proof is by induction on the size of φ. We omit the subscript M since it is clear from the context. If $\varphi = p$ or $\varphi = \bot$ the result is immediate since $V(s) = V^{\subset}(s)$ for any state s. Suppose that the statement of the lemma holds for any formula of size $k \leq n$, and let φ be a formula of size $n + 1$. If the main connective of φ is a boolean connective, the result follows by the induction hypothesis. If $\varphi = \Diamond_a \psi$:

($\Rightarrow$) $M, s \models \Diamond_a \psi$ iff there is s' such that $e = (s, a, s') \in R$ and $M, s' \models \psi$. By induction hypothesis $M^{\subset}, s' \models (\psi)^{\star}$. By the definition of $M^{\subset}$ we have that $(s, n_e, s') \in R_{n_e}$ thus s and s' are adjacent, with respect to R_{n_e}, in $M^{\subset}$. It follows that $M^{\subset}, s \models \Diamond_{n_e} (\psi)^{\star}$. But this means that $M^{\subset}, s \models \bigvee_{e \in R_a} \Diamond_{n_e} (\psi)^{\star}$ as we wanted.

($\Leftarrow$) $M^{\subset}, s \models (\Diamond_a \psi)^{\star}$ iff $M^{\subset}, s \models \bigvee_{e \in R_a} \Diamond_{n_e} (\psi)^{\star}$. If this is the case, then there is at least an edge $e = (s, a, s') \in R_a$ such that $M^{\subset}, s \models \Diamond_{n_e} (\psi)^{\star}$ which implies that $M^{\subset}, s' \models (\psi)^{\star}$. By induction hypothesis $M, s' \models \psi$. Since, by definition of $M^{\subset}$, $(s, a, s') \in R$ in M, we get the result.

If $\varphi = \blacklozenge^{\subset}_a \psi$:

($\Rightarrow$) $M, s \models \blacklozenge^{\subset}_a \psi$ iff there is a non-empty subset $E = \{e_1, \ldots e_n\}$ of R_a such that $M \setminus E, s \models \psi$. By induction hypothesis $(M \setminus E)^{\subset}, s \models (\psi)^{\star}$. As stated in Remark 2, $(M \setminus E)^{\subset} = M^{\subset} \setminus \{n_{e_1}, \ldots, n_{e_n}\}$. Since $M^{\subset} \setminus \{n_{e_1}, \ldots, n_{e_n}\}, s \models (\psi)^{\star}$ we deduce that $(\cdots (M^{\subset}, \setminus \{n_{e_1}\}) \setminus \cdots) \setminus \{n_{e_n}\}, s \models \psi^{\star}$, thus $M^{\subset}, s \models \blacklozenge_{n_E} \psi^{\star}$.

($\Leftarrow$) $M^{\subset}, s \models (\blacklozenge^{\subset}_a \psi)^{\star}$ iff $M^{\subset}, s \models \bigvee_{E \in 2^{R_a} \setminus \emptyset} (\blacklozenge_{n_E} (\psi)^{\star})$. This means that there is a subset E of R_a such that $E = \{e_1, \ldots, e_n\}$ and $M^{\subset}, s \models \blacklozenge_{n_E} (\psi)^{\star}$. By definition of satisfaction, this means that $(\cdots (M^{\subset} \setminus \{n_{e_1}\}) \setminus \cdots) \setminus \{n_{e_n}\} \models (\psi)^{\star}$, thus $M^{\subset} \setminus \{n_{e_1}, \ldots, n_{e_n}\}, s \models (\varphi)^{\star}$. As stated in Remark 2, this implies that $(M \setminus E)^{\subset}, s \models (\psi)^{\star}$. By induction hypothesis, $M \setminus E, s \models \psi$. By the fact that each $e_i \in R_a$, we obtain that $M, s \models \blacklozenge^{\subset}_a \psi$ as we wanted.

From the above lemma and the fact the model checking problem is decidable for SML, we immediately deduce the following theorem.

Theorem 2. *The model checking problem for SSML is decidable: if $M = \langle S, s_0, \Sigma, R, V\rangle$ is a finite rooted Kripke Structure and φ an SSML formula, we can decide whether $M \models \varphi$ or not.*

It should not be surprising that we can express the existence of Attacker winning strategies for our games by using SSML: we have designed such logic with precisely this goal in mind.

Let $AG = \langle S, s_0, \Sigma, R, V, T\rangle$ be an Attack Graph and Δ a non-empty subset of Σ. If φ is an SSML formula, we define the two SSML-formulas:

$$
\blacksquare^{\subset}_{\Delta} \varphi \doteq \bigwedge_{a \in \Delta} \blacksquare^{\subset}_a \varphi \qquad \Diamond \varphi \doteq \bigvee_{a \in \Sigma} \Diamond_a \varphi
$$

A strategy is a plan of action. As it is logical, the plan is winning when it leads me to victory, whatever my opponent's plan of action. Thus, a winning strategy can be expressed as an alternance of universally quantified sentences and existentially quantified sentences "for all actions of my Opponent, there is an action that I can make that leads me to victory". Let us put ourselves in the villain's shoes: suppose that we are the Attacker, and that, by playing, we have reached a certain state s of an Attack Graph AG. It is now Defender's turn. If I have a winning strategy, I must be able to reach a successor state s' of s in whatever subgraph AG' of AG that is Δ-reachable from AG. Said differently, we must have that $AG, s \models \blacksquare^{\subset}_{\Delta} \Diamond \varphi = (\blacksquare^{\subset}_{a} \Diamond\varphi) \wedge \cdots \wedge (\blacksquare^{\subset}_{b} \Diamond \varphi)$ for some formula φ that expresses the winning condition. Such formula is nothing but the SSML version of the one we have defined in 1.

Definition 12 (Winning Formulas)**.** *The family $\{\psi^n_\Delta\}_{n\in\mathbb{N}}$ of Winning formulas is defined by induction on n as follows:*

$$\psi^n_\Delta = \begin{cases} \mathsf{win} & \text{if } n = 0 \\ \Diamond \top \wedge \blacksquare^{\subset}_{\Delta} \Diamond(\psi^{n-1}_\Delta \vee \mathsf{win}) & \text{otherwise} \end{cases} \quad (2)$$

We are now ready to prove the main result of our paper. Namely, that the existence of a winning Attacker Δ-strategy over an Attack Graph AG is equivalent to the truth of a winning formula over AG.

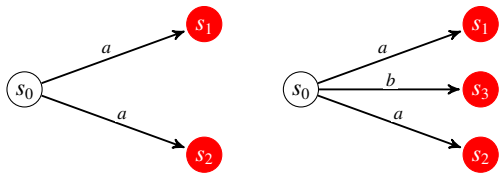
Theorem 3. *For any Attack Graph $AG = \langle S, s_0, \Sigma, R, V, T\rangle$ for any non-empty subset Δ of Σ, if $|S| = n$, then $AG, s_0 \models \mathsf{win} \vee \psi^{n-1}_\Delta$ iff there is a winning Δ-strategy over AG for the Attacker.*

Both directions of the theorem are proved by induction on n.

($\Rightarrow$) An Attack Graph has a non-empty set of states by definition. Thus, the base case is $n = 1$. $AG, s_0 \models \mathsf{win} \vee \mathsf{win} \iff T = \{s_0\}$. The strategy we are looking for is the path AGs_0. Suppose that the statement of the theorem holds for any Attack Graph AG' of size $k \leq n$, and let AG be an Attack Graph of size $n + 1$. Suppose that $AG, s_0 \models \mathsf{win} \vee \psi^n_\Delta$. Without loss of generality, we can suppose that $AG, s_0 \not\models \mathsf{win}$ because otherwise the result is trivial. Thus $AG, s_0 \models \psi^n_\Delta$. This means that $AG, s_0 \models \Diamond \top \wedge (\blacksquare^{\subset}_{\Delta} \Diamond (\psi^{n-1}_\Delta \vee \mathsf{win}))$. By the definition of satisfaction, $AG, s_0 \models (\Diamond \top \wedge \blacksquare^{\subset}_{a} \Diamond (\psi^{n-1}_\Delta \vee \mathsf{win}))$ for any $a \in \Delta$. Let a one of the element of Δ. We have that, $AG, s_0 \models \Diamond \top \wedge \blacksquare^{\subset}_{a} \Diamond (\psi^{n-1}_\Delta \vee \mathsf{win})) \iff AG, s_0 \models \Diamond\top$ and $AG, s_o \models \blacksquare^{\subset}_{a} \Diamond (\psi^{n-1}_\Delta \vee \mathsf{win}))$. We can thus conclude that there is a $s' \neq s$ such that $(s, c, s') \in R$ for some $c \neq a$ and $s' \models \psi^{n-1}_\Delta \vee \mathsf{win}$. Consider the subgraph AG_a of AG obtained by: erasing the vertex s_0; erasing any edge that has s_0 as source or target; erasing any edge whose label is a and, in which the initial state is s'. The size of a such graph is n and $AG_a, s' \models \psi^{n-1}_\Delta \vee \mathsf{win}$. By induction hypothesis, there is a winning Δ-strategy $\mathcal{S}_a$ over AG_a. We obtain a winning strategy $\mathcal{S}$ for AG by taking the sequence $AG\, s_0$ and putting an edge from s_0 to the root of $\mathcal{S}_a$ for any $a \in \Delta$.

($\Leftarrow$) If $n = 1$ and there is a winning Attacker strategy $\mathcal{S}$ over AG, then $\mathcal{S} = \{AG\, s_0\}$. Thus $s_0 \in T$ and this means that $AG, s_0 \models \mathsf{win}$ and we can conclude. Suppose that the statement of the theorem holds for any Attack Graph AG' of size $k \leq n$, and let AG be an Attack Graph of size $n + 1$. Let $\mathcal{S}$ be any winning strategy over AG. Suppose that $\mathcal{S}$ contains more than one play (otherwise the result is trivial). By Proposition 1, we can suppose that $\mathcal{S}$ is stubborn. Let $Next = \{s_i \in S \mid \exists\rho\ \rho \text{ is a branch of } \mathcal{S} \wedge \rho_3 = s_i\}$ i.e., any $s_i \in Next$ is a state that is adjacent to s_0 in the subgraph of AG obtained by erasing all edges labeled by b for some $b \in \Delta$. Remark that $Next$ is finite and non-empty and that, since $\mathcal{S}$ is stubborn, each s_i is different from s_0. For any s_i, let AG_{s_i} be the subgraph of AG obtained by erasing s_0, any edge having s_0 as source and whose initial state is s_i. Define $\mathcal{S}_{s_i}$ to be the tree whose branches are of the form $AG_{s_i}\, s_i \rho'$ for $s_i \rho'$ subsequence of a play of $\mathcal{S}$. Clearly, $\mathcal{S}_{s_i}$ is a winning Attacker Strategy over the Attack Graph AG_{s_i} for any s_i. Since AG_{s_i} has n-states, we can use the induction hypothesis and conclude that $AG_{s_i}, s_i \models \mathsf{win} \vee \psi^{n-1}_\Delta$. Any $\mathcal{S}_{s_i}$ is stubborn, thus s_0 does not appear in any of its play. We deduce that for any $s_i \in Next$, $AG, s_i \models \mathsf{win} \vee \psi^{n-1}_\Delta$. From this latter fact, and by the definition of $Next$ we conclude that $AG, s_0 \models \Diamond \top \wedge \blacksquare^{\subset}_{\Delta} \Diamond(\psi^{n-1}_\Delta \vee \mathsf{win})$.

Example 3. *Consider the two following Attack Graphs:*



There is not $\{a\}$*-winning strategy over the Attack Graph on the left. In fact, it does not satisfy the formula* $\mathsf{win} \vee (\Diamond\top \wedge \blacksquare_a^{\subset} \Diamond((\Diamond \top \wedge \blacksquare_a^{\subset} \Diamond \mathsf{win}) \vee \mathsf{win}))$*: we have that* $s_0 \not\models \mathsf{win}$ *and* $s_0 \not\models \blacksquare_a^{\subset} \Diamond((\Diamond \top \wedge \blacksquare_a^{\subset} \Diamond \mathsf{win}) \vee \mathsf{win}))$*. If we erase all edges labeled by the letter a, no edges are left. Thus* $s_0 \not\models \Diamond\varphi$ *for any* φ*.*

Consider the Attack Graph on the right, call it AG_r *and let* AG_r^a *be* AG_r *without edges labeled by a and* AG_r^b *be the* AG_r *without the edge labeled by b. The formula* $\mathsf{win} \vee \psi_3^{\{a,b\}}$ *is satisfied by* AG_r*, and there is a winning strategy* $\mathcal{S}$ *whose branches are:* $\rho = AG_r\, s_o\, AG_r^a\, s_3$ *and* $\tau = AG_r\, s_0\, AG_r^b\, s_1$

7. Conclusion and Future Work

We have presented a natural class of two-player games over Attack Graphs. Such games are played by an Attacker and a Defender. The Attacker tries to reach some vertex of the Attack Graph, while the Defender tries to prevent him from doing so. To do this, the Defender can eliminate subsets of arcs in the graph. We have seen how these games can be viewed as a generalized version of sabotage games, we have formally defined plays of such games and winning strategies. Finally, we have introduced a variant of Sabotage Modal Logic, showed that the model checking problem for such logic is decidable and that we can express the existence of a winning strategy for our subset sabotage games by formulas of the new logic.

The games we have defined are perfect information games; both players know, at every point in the game, the location of the other player. This assumption is unrealistic: during a cyberattack, a possible Defender may not know what state an Attacker is in, and conversely, an Attacker may not be aware of changes made by the Defender to counter his attack. We would therefore like to extend our play model in order to include this type of imperfect information. From the Defender's point of view, this could be implemented as an equivalence class between Attack Graph states. From the Attacker's point of view, on the other hand, we could think of a notion of weak bisimulation between Attack Graphs: the Attacker considers as equal two models that are bisimilar up to the identification of some subset of arcs.

We have shown that the model-checking problem for SSML logic is decidable. However, we have not investigated the complexity of that problem (which must, however, be at least P-Space). We leave this investigation for future work. We suspect that a bisimulation notion for SSML logic can be obtained by slightly modifying the one for SML and that, at the same, a complete proof system for SSML can be obtained in terms of tableaux. In conclusion, we suspect that the satisfiability problem for SSML logic is undecidable. Indeed, the same problem is undecidable for SML and since our logic is SML in which we quantify over subset of arcs of a graph, our intuition tells us that the satisfiability problem for SSML can only be more difficult than the one of SML.

Finally, we can consider to extend logics for the strategic reasoning such as ATL [1] and Strategy Logic [25] by capturing the features of the sabotage modalities $\blacklozenge$ and $\blacklozenge^{\subset}$. In this way, we can gain expressive power and check whether the attacker has a strategy to win the game via strategic operators. Furthermore, in this context, we can see $\blacklozenge^{\subset}$ as a graded modality of $\blacklozenge$ as done in logics for strategies [2,13]. However, as we mentioned earlier the more realistic setting for games is with imperfect information, but unfortunately, the model checking problem with imperfect information for strategic logics is undecidable in general [10]. Given the relevance of this setting, even partial solutions to the problem can be useful, such as abstractions either on the information [4,6] or on the strategies [5] or on the formulas [14]. In conclusion, we can embed the mentioned techniques to provide a more powerful framework.

References

[1] R. Alur, T. Henzinger, and O. Kupferman. Alternating-Time Temporal Logic. *Journal of the ACM*, 49(5):672–713, 2002.

[2] B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded modalities in strategy logic. *Inf. Comput.*, 261:634–649, 2018.

[3] G. Aucher, J. V. Benthem, and D. Grossi. Modal logics of sabotage revisited. *Journal of Logic and Computation*, 28(2):269 – 303, Mar. 2018.

[4] F. Belardinelli, A. Lomuscio, and V. Malvone. An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Con-*

ference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019, pages 6030–6037. AAAI Press, 2019.

[5] F. Belardinelli, A. Lomuscio, V. Malvone, and E. Yu. Approximating perfect recall when model checking strategic abilities: Theory and applications. *J. Artif. Intell. Res.*, 73:897–932, 2022.

[6] F. Belardinelli and V. Malvone. A three-valued approach to strategic abilities under imperfect information. In D. Calvanese, E. Erdem, and M. Thielscher, editors, *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pages 89–98, 2020.

[7] D. Catta, J. Leneutre, and V. Malvone. Subset sabotage games & attack graphs. In A. Ferrando and V. Mascardi, editors, *Proceedings of the 23rd Workshop "From Objects to Agents", Genova, Italy, September 1-3, 2022*, volume 3261 of *CEUR Workshop Proceedings*, pages 209–218. CEUR-WS.org, 2022.

[8] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In O. Grumberg and H. Veith, editors, *25 Years of Model Checking - History, Achievements, Perspectives*, volume 5000 of *Lecture Notes in Computer Science*, pages 196–215. Springer, 2008.

[9] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[10] C. Dima and F. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[11] K. Durkota, V. Lisý, B. Bosanský, and C. Kiekintveld. Approximate solutions for attack graph games with imperfect information. In M. H. R. Khouzani, E. A. Panaousis, and G. Theodorakopoulos, editors, *Decision and Game Theory for Security - 6th International Conference, GameSec 2015, London, UK, November 4-5, 2015, Proceedings*, volume 9406 of *Lecture Notes in Computer Science*, pages 228–249. Springer, 2015.

[12] K. Durkota, V. Lisy, B. Bošansky, and C. Kiekintveld. Optimal network security hardening using attack graph games. IJCAI'15, page 526–532. AAAI Press, 2015.

[13] M. Faella, M. Napoli, and M. Parente. Graded alternating-time temporal logic. *Fundam. Informaticae*, 105(1-2):189–210, 2010.

[14] A. Ferrando and V. Malvone. Towards the combination of model checking and runtime verification on multi-agent systems. In F. Dignum, P. Mathieu, J. M. Corchado, and F. de la Prieta, editors, *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complex Systems Simulation. The PAAMS Collection - 20th International Conference, PAAMS 2022, L'Aquila, Italy, July 13-15, 2022, Proceedings*, volume 13616 of *Lecture Notes in Computer Science*, pages 140–152. Springer, 2022.

[15] T. Heberlein, M. Bishop, E. Ceesay, M. Danforth, C. Senthilkumar, and T. Stallard. A taxonomy for comparing attack-graph approaches. *[Online] http://netsq. com/Documents/AttackGraphPaper. pdf*, 2012.

[16] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 121–130, 2006.

[17] W. Jamroga and A. Murano. Module checking of strategic ability. In G. Weiss, P. Yolum, R. H. Bordini, and E. Elkind, editors, *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015, Istanbul, Turkey, May 4-8, 2015*, pages 227–235. ACM, 2015.

[18] N. R. Jennings and M. Wooldridge. Application of intelligent agents. In *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, 1998.

[19] K. Kaynar. A taxonomy for attack graph generation and usage in network security. *J. Inf. Secur. Appl.*, 29(C):27–56, Aug. 2016.

[20] O. Kupferman, M. Vardi, and P. Wolper. An Automata Theoretic Approach to Branching-Time ModelChecking. *Journal of the ACM*, 47(2):312–360, 2000.

[21] O. Kupferman, M. Vardi, and P. Wolper. Module Checking. *Information and Computation*, 164(2):322–344, 2001.

[22] R. P. Lippmann and K. W. Ingols. An annotated review of past papers on attack graphs. 2005.

[23] C. Löding and P. Rohde. Model checking and satisfiability for sabotage modal logic. In P. K. Pandya and J. Radhakrishnan, editors, *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, volume 2914 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 2003.

[24] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of the 21th International Conference on Computer Aided Verification (CAV09)*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, 2009.

[25] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions in Computational Logic*, 15(4):34:1–34:47, 2014.

[26] T. H. Nguyen, M. Wright, M. P. Wellman, and S. Baveja. Multi-stage attack graph security games: Heuristic strategies, with empirical game-theoretic analysis. MTD '17, page 87–97, New York, NY, USA, 2017. Association for Computing Machinery.

[27] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC '03, page 86, USA, 2003. IEEE Computer Society.

[28] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345, 2006.

[29] C. Phillips and L. P. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the 1998 workshop on New security paradigms*, pages 71–79, 1998.

[30] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. pages 273–284, 02 2002.

[31] J. van Benthem. *An Essay on Sabotage and Obstruction*, pages 268–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[32] Y. Zhang and P. Malacaria. Bayesian stackelberg games for cyber-security decision support. *Decis. Support Syst.*, 148:113599, 2021.