Network Protocol Safe Configuration Search in One Shot

Northwestern University

You Li, Kaiyu Hou, Hai Zhou, Yan Chen

Introduction

A large number of works done by networking security researchers use model checking. Each of the works comes with one or more formal models. Nevertheless, we find these works do not fully exploit the potentials of their models.

Traditionally, researchers select a set of correctness properties the protocol or system should follow. If the model checker determines the properties are satisfied, one can conclude the protocol or system is safe with regard to the properties. Often times, the researchers expect to find counterexamples, which suggest possible traces of attacks. Regardless of their purposes, researchers always choose a designated initial state, so their results only apply to that specific initial state.

The above problem is a *decision problem*, given the model, the initial state, and the properties. However, networking protocols and systems are highly complicated. There usually exist more than one way to deploy. In that regard, we propose to convert the decision problem into a *search problem* given the model and the properties; the result is a set of states which satisfy the properties.

We call that set of states *secure option space*. It has two-fold benefits. For designers, they want to know in what circumstances their designs are secure, especially when a design has multiple asynchronized components.For deployment engineers, they want to configure the systems to meet all security criteria and make sure the systems always execute as expected, provided there are so many configurable parameters and initial states for a network system. In either case, no one will want to enumerate all possible initial states, and run model checker for each of them.

Preliminaries

A protocol or system can be formalized as a finite state transition system $S : (\bar{i}, \bar{x}, I(\bar{x}), T(\bar{i}, \bar{x}, \bar{x'}))$ consisting of input variables $\bar{i}$, state variables $\bar{x}$, initial state $I(\bar{x})$, and a transition relation $T(\bar{i}, \bar{x}, \bar{x'})$: the function of inputs $\bar{i}$ and current state $\bar{x}$ to determine the next state $\bar{x'}$. A state s is an assignment of values to all state variables $\bar{x}$. A state either satisfies a correctness property: $s \vDash P$, or falsifies it: $s \nvDash P$.

Correctness properties can be divided into two categories: **safety properties** and **liveness properties**.

Secure Option Space Search for Safety Properties

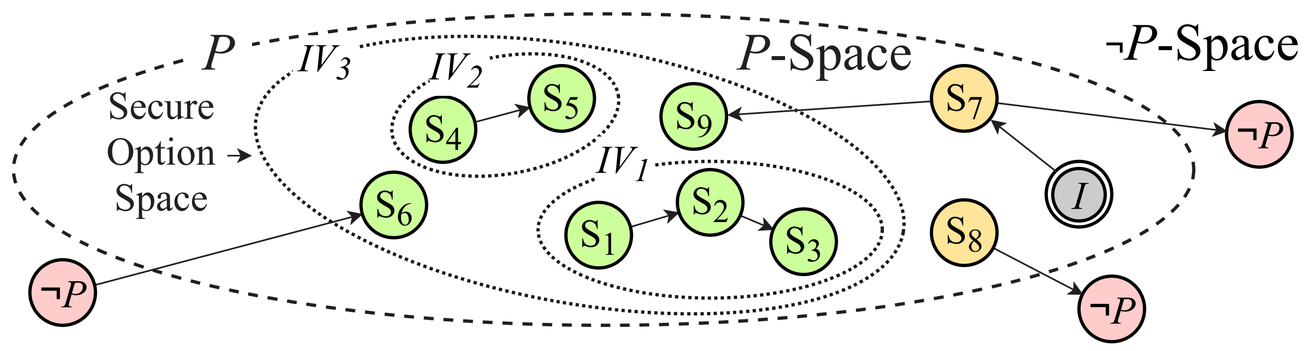

Figure 1. An illustrative example of searching the secure option space for the safety property P. The original initial state I lies in the P-space, but it is not secure as it can reach the $\neg P$-space. The greatest inductive invariant, IV_3, is the secure option space we aim to find. Any state lies in IV_3 is always guaranteed to satisfy P. Note there exist other inductive invariants like IV_1, IV_2, $IV_1 \cup IV_2$, *etc.*

A safety property P states that bad things can *never* happen. For example, an unauthorized party should not be allowed to broadcast fake alerts by abusing the wireless alert system. On a formal model, it is equivalent to say that none of the bad states ($\neg P$-states) can be reached. Therefore, the problem of searching a secure option space becomes searching for states which cannot lead to any bad states.

An assertion F is an *inductive invariant* if $F \wedge T \Rightarrow F'$: there is no outgoing edges from F-states to $\neg F$-states. Here is the key insight: if there exists an inductive invariant F^* which does not intersect with the $\neg P$-space, all F^*-states are secure (Figure 1).

IC3 is the *state-of-the-art* symbolic model checking algorithm. It maintains a sequence of frames $F_1 \cdots F_k$, in which $F_i \wedge T \Rightarrow F'_{i+1}$. Our search for the secure option space can benefit from the design of the IC3: *i)* It will always return an inductive invariant F^* if exists. *ii)* After every iteration, a new frame $F_{k+1} \leftarrow P$ is produced. Since only P-states can constitute the secure option space, the search for the secure option space always starts from the largest possible candidate.

Secure Option Space Search for Liveness Properties

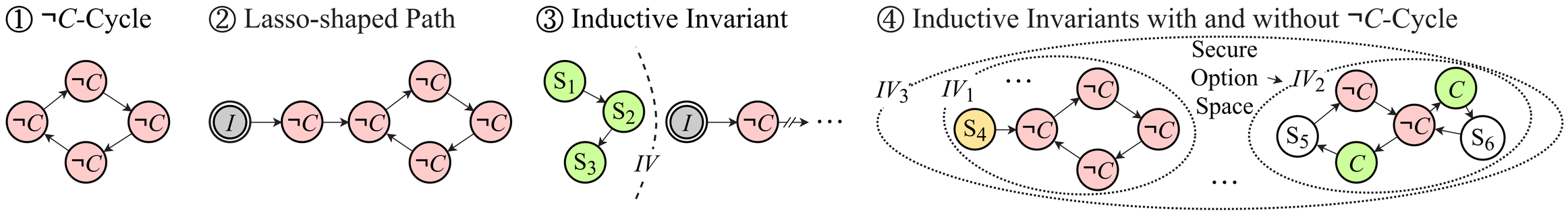

Figure 2. ① A $\neg C$ cycle, in which the condition C of the liveness property P is never satisfied. ② The system is not secure as there exists a path from the original initial state I to the $\neg C$-cycle. ③ Any potential $\neg C$-cycles outside of the inductive invariant IV have no effect to the states within IV. ④ IV_3 and IV_1 have a $\neg C$ cycle in them, so they cannot be determined to be secure. IV_2 is a secure option space we aim to find.

A liveness property states that good things should *eventually* happen. For instance, a phone call to an emergency number should be eventually routed to a public safety center. No livenesses can be satisfied if a system does not move forward; so livenesses are usually used in conjunction with fairness constraints.

Liveness properties can be refuted if there exists a *lasso-shaped* path starting from an initial state I, and none of the states along that path satisfies the condition of the liveness properties. A lasso-shaped path from I is a path that has a cycle (Figure 2 ①) and a connection from I to any state on the cycle (Figure 2 ②).

FAIR is an algorithm to find such paths we are looking for. It first queries the SAT solver for a group of states that satisfies a certain fairness condition. Then it queries IC3 for a connection from the initial state to one of the states in the group. If that succeeds, the algorithm queries IC3 for multiple times to build a cycle with the group of states. If all steps above succeed, FAIR finds a lasso-shaped counterexample. However, if any of the queries to IC3 fail, FAIR accumulates a new inductive invariant returned by IC3. Notice that no cycle can go through a boundary depicted by an inductive invariant.

In order to find the secure option space with regard to liveness properties, we require the fairness condition to be that no states satisfy the condition C of the liveness properties. That fairness condition applies to both the group-of-state queries and the connection queries. Then we can reach a key observation: if there exists no such cycle within an inductive invariant, every state within that inductive invariant belongs to the secure option space. (Figure 2 ③ and Figure 2 ④)

Conclusion

We have implemented our algorithms for safety and liveness properties and tested them with synthetic models. Given any states, a SAT solver can determine in one shot if they reside in a found secure option space. In the past, researchers make efforts to find vulnerabilities within systems. Then they mitigate those vulnerabilities, hoping the systems can become more secure. Such a process could iterate forever. Our proposed method deals with this problem from the reverse direction: efficiently searching for the states and configurations which must be secure. Protocol designers, deployment engineers could benefit from the method.

@ ACM SIGCOMM 2020 Posters & Demos