

Network Protocol Safe Configuration Search in One Shot

You Li, Kaiyu Hou, Hai Zhou, Yan Chen
Northwestern University, USA

CCS CONCEPTS

• **Networks** → Protocol correctness; Protocol testing and verification; • **Security and privacy** → Logic and verification.

KEYWORDS

Network system configuration; Model checking; Protocol security

1 INTRODUCTION

Model checking techniques can verify correctness properties by exhaustively traversing the state space of a formal model. A large number of works [1, 4, 5] done by networking researchers use model checking. Each of the works comes with one or more formal models, which constitutes a valuable asset for the networking security research community. Nevertheless, we find these works do not fully exploit the potentials of their models.

A better way to extract more underlying information from those models is to search for the *secure option space*: given a set of correctness properties, if the model executes from any state within the secure option space, it is guaranteed these properties always hold. Traditional model checkers cannot solve it, as their targeted goal is to decide whether an instance of the model is safe with regard to the given properties. Leveraging recent breakthroughs of symbolic model checkers, we devise algorithms to search for the secure option space. Moreover, the found results can be stored compactly and queried efficiently. We have implemented prototypes of our proposed algorithms and are testing them on existing models of cellular network protocols.

2 MOTIVATION

A network 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: $s \models P$, or falsifies it: $s \not\models P$.

Correctness properties can be divided into two categories: safety properties and liveness properties. Safety properties state that some bad things can never happen, while liveness properties state that some good things should eventually happen.

Traditionally, after the model is specified, researchers select a set of correctness properties the original protocol or system should follow. If the model checker determines the properties are satisfied,

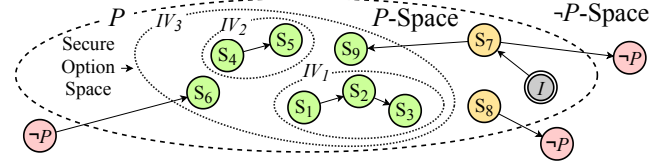


Figure 1: An illustrative example of searching the secure option space for the safety property P . The original initial state I lies in the P -space, but it is not secure as it can reach the $\neg P$ -space. The greatest inductive invariant, IV_3 , is the secure option space we aim to find. Any state 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.

one can conclude the original protocol or system is safe with regard to the properties. In the field of networking security, often times the researchers expect to find counterexamples, because counterexamples suggest possible traces of attacks or other vulnerabilities. Regardless of their purposes and methodologies, researchers always choose a designated initial state, so their results only apply to that specific initial state.

In abstract, the above problem is a decision problem given the model, the initial state, and the correctness properties; the result is either satisfy or falsify. Network protocols and systems are highly complicated. There usually exist more than one way to deploy a single network system. 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*. The benefits of finding the secure option space are two-fold. For designers, they want to know in what circumstances their designs are secure, especially when a design has multiple asynchronized components, keeps evolving over time, and has more than one contributing team. It is desired by designers that the secure option space covers all use cases and customer requirements. 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.

We describe two symbolic model checking algorithms to search secure option spaces for safety and liveness properties, respectively.

3 SECURE OPTION SPACE SEARCH FOR SAFETY PROPERTIES

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:

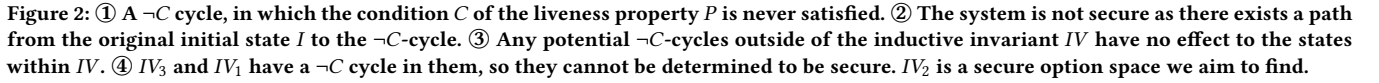
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCOMM '20 Demos and Posters, August 10–14, 2020, Virtual Event, USA

© 2020 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8048-5/20/08.

<https://doi.org/10.1145/3405837.3411377>



Once an inductive invariant is found, we can expand its size recursively by setting the found F^* as F_0 in the next execution of the above algorithm to find a larger secure option space.

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).

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, as well as researchers, could benefit from the method.

REFERENCES

- [1] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A formal analysis of 5G authentication. In *2018 ACM SIGSAC Conference on Computer and Communications Security*.
- [2] Aaron Bradley. 2011. SAT-based model checking without unrolling. In *2011 International Workshop on Verification, Model Checking, and Abstract Interpretation (VMCAI)*. Springer.
- [3] Aaron Bradley, Fabio Somenzi, Ziad Hassan, and Yan Zhang. 2011. An incremental approach to model checking progress properties. In *2011 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE.
- [4] Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. 2018. LTEInspector: A systematic approach for adversarial testing of 4G LTE. In *the 25th Network and Distributed Systems Security (NDSS) Symposium*.
- [5] Yinbo Yu, You Li, Kaiyu Hou, Yan Chen, Hai Zhou, and Jianfeng Yang. 2019. CellScope: Automatically Specifying and Verifying Cellular Network Protocols. In *2019 SIGCOMM Conference Posters and Demos*. ACM.