Poster: Securing Distributed System Configuration through Optimization and Reasoning on Graphs

Hamed Soroush
Palo Alto Research Center
hsoroush@parc.com

Shantanu Rane
Palo Alto Research Center
srane@parc.com

Massimiliano Albanese
George Mason University
malbanes@gmu.edu

***Abstract*—Complex distributed systems are inherently difficult to secure due to the many interdependencies amongst their components, vulnerabilities, and configuration parameters. To address this problem, we present an approach for improving the security posture of distributed systems by examining the security impact of configuration changes across their interdependent components. We construct a graph-based model of the system and its vulnerabilities and use it to analyze the attack surface and the impact of attacks. We show how the model can be optimized using SMT solvers to derive configurations that minimize the impact of attacks while preserving system functionality.**

I. Introduction

As cyber-systems become more distributed, connected, and complex, configuration analytics is beginning to play an increasingly critical role in their correct and secure operation. Attackers typically rely on unpatched vulnerabilities and configuration errors to gain unauthorized access to system resources. Misconfigurations can occur at any level of a system's architecture and correctly configuring systems becomes increasingly complex when multiple interconnected systems are involved. *Security Misconfiguration* was listed by OWASP amongst the ten most critical web application security risks in 2017 [1]. Fixing such misconfigurations requires administrators to take into account the security implications of configuration changes on the entire system, going beyond fine-tuning parameters of individual components. Given the increasingly large scale of cyber-systems, this task must be automated to the maximum extent possible. Previous work on handling configuration errors largely ignores the security impact of configurations of connected components.

To address these challenges we propose (i) a method to integrate individual configurations into a graph-based model to capture within-component and between-component dependencies among configuration parameters; (ii) algorithms to efficiently and automatically identify configurations that minimize the attack surface, and more importantly, potential attack impact, while maintaining functionality and performance. Our system also provides visual, human-understandable evidence for the optimality of the selected configuration set. In the following, we describe construction of the model and our reasoning methodology along with preliminary results.

II. Sample System

We use the sample system depicted in Figure 1 to describe our technical approach. The system comprises an Unmanned Aerial Vehicle (UAV), an Unmanned Ground Vehicle (UGV), and a backend mission control station. During a mission, the UAV requires both land and air route processing information from the mission control unit. The autonomous vehicle only requires land route processing information to operate.

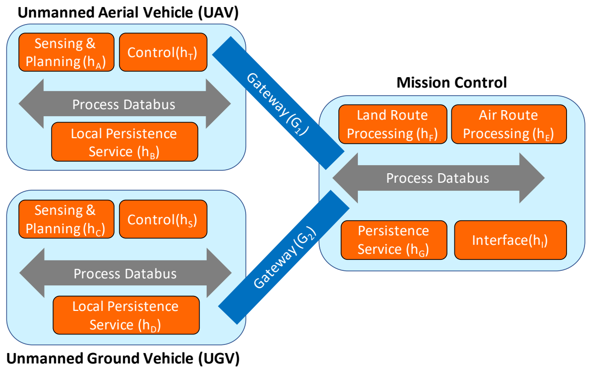

Unmanned Aerial Vehicle (UAV)
Sensing & Planning (h_A)
Control(h_T)
Process Databus
Local Persistence Service (h_B)
Gateway(G_1)
Mission Control
Land Route Processing (h_F)
Air Route Processing (h_E)
Process Databus
Persistence Service (h_G)
Interface(h_I)
Gateway(G_2)
Sensing & Planning (h_C)
Control(h_S)
Process Databus
Local Persistence Service (h_D)
Unmanned Ground Vehicle (UGV)

Fig. 1: Sample system used to illustrate our approach

III. Model Construction

Our proposed framework extends [2] by using configuration information about individual components to construct a three-layer directed graph encoding all the information needed to compute optimal configurations. This graph is comprised of the three interconnected directed subgraphs described below.

The *dependency subgraph* models the functional dependencies between components of a complex system. In this subgraph, each vertex represents a functional component, and is labeled with a utility value and one of several dependency types. Each edge in the dependency subgraph represents a functional dependency on another component, as specified by the dependency label of the parent vertex. The *attack subgraph* models the propagation of potential multi-step attacks through the system. In the attack subgraph, each vertex represents a vulnerability, and an edge indicates that exploiting the parent vulnerability would set the stage for the attacker to exploit the child vulnerability. Each edge is also labeled with the probability of the attack progressing along that edge. Finally, the *configuration subgraph* models the relationships between configuration parameters, both within individual components and across components of a complex system. There are two classes of vertices in this subgraph: *Class 1* vertices capture per-component configuration parameters; *Class 2* vertices capture constraints (or conditions) on configuration parameters. These constraints are defined by functional system requirements and admissibility of configuration settings. Furthermore, some of these constraints or conditions may enable or disable preconditions for system vulnerabilities, inducing a particular attack subgraph for that configuration.

The three subgraphs are connected with 3 types of edges, giving the complete model of the system, as shown in Figure 2. A directed edge from a component in the dependency graph to a Class 1 vertex in the configuration graph defines the configuration parameters associated with that component. A directed edge from a Class 2 node in the configuration subgraph to a vertex in the attack subgraph (i.e., a vulnerability) implies that the condition represented by the Class 2 vertex is a precondition for that vulnerability. A directed edge from a vertex in the attack subgraph to a vertex in the dependency subgraph (i.e., a system component) indicates that exploiting that vulnerability impacts the component by an amount proportional to the exposure factor labeling the edge.

IV. Analysis and Approach

To find configurations that measurably reduce the impact of attacks, information captured in the three subgraphs discussed above must be efficiently and jointly analyzed. In fact, globally optimal security decisions (e.g., deciding which vulnerability to patch or make unreachable through configuration changes) cannot be made without dependency information. To illustrate, consider the 3-layer graph model of our reference system in Fig. 2. Suppose that an attacker exploits vulnerability v_C. This makes h_C completely unavailable, as the exposure factor is 1. As h_T strictly depends on h_C, h_T also becomes unavailable, leading to a marginal impact of $7+7=14$ as a consequence of exploiting v_C. In this example, we assume a simple impact function, $impact(v_j) = \sum_{h \in H}(s_{j-1}(h) - s_j(h)) \cdot u(h)$, where $s_{j-1}(h)$ and $s_j(h)$ respectively denote the relative residual utility of asset h before and after exploitation of vulnerability v_j in an attack path $P = (v_1, \ldots, v_n)$, and $u(h)$ is the original utility of h. For a given attack step v_j, this impact function sums the marginal losses for all the components affected (either directly or indirectly) by the exploitation of v_j. After exploiting v_C, the attacker may take one of two steps, exploiting either v_D with probability 0.7 or v_F with probability 0.3. Intuition suggests that, as the attacker is more likely to exploit v_D, that vulnerability should be patched or addressed before v_F. However, this approach turns out to be inefficient, as we now explain: the additional impact of exploiting v_D would be $0.7 \cdot 5 = 3.5$, as h_C and h_T are already unavailable because of the previous exploit; instead, the additional impact of exploiting v_F would be $0.7 \cdot 7 + 8 + 10 = 22.9$, as compromising h_F also makes h_A and h_S unavailable.

Formally, the impact of the adversary sequentially exploiting the vulnerabilities $v_1, \ldots, v_n$ in a given path $P = (v_1, \ldots, v_n)$ in the attack subgraph can be computed as:

$$impact(P) = \sum_{j=1}^{n} \sum_{h \in H} (s_{j-1}(h) - s_j(h)) \cdot u(h) \qquad (1)$$

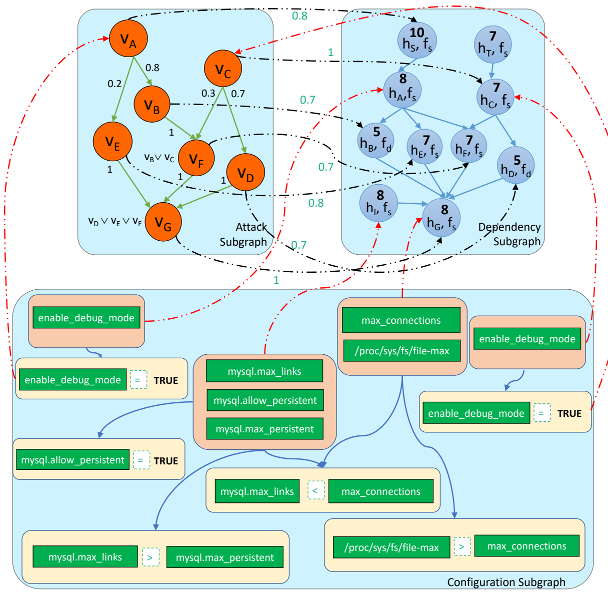


Fig. 2: 3-layer directed graph model of system in Fig. 1

We created a reasoning framework that uses an SMT solver to solve the following problem and find configurations that minimize the attack impact while preserving functionality and satisfying configuration constraints:

Find configuration $F^* = (f_1^*, f_2^*, \ldots, f_k^*)$ such that:

1) Configuration subgraph constraints are satisfied
2) Dependency subgraph constraints are satisfied
3) $F^* = \arg\min_{F} \sum_{P \in A(F)} impact(P)$

where $P = (v_1, \ldots, v_n)$ is any path in the attack subgraph $A(F)$ induced by the configuration F.

For our example in Fig. 2, the solver determines that debug_mode must be set to false for both h_A and h_C. If no solution is found, we relax the constraints starting from the unsatisfiable core and search again for a limited number of iterations.

V. Conclusions and Future Work

We implemented a preliminary prototype of the proposed framework, using custom procedures in the Neo4j graph database for modeling and automatically deriving constraints. We used the Z3 SMT solver to find solutions that satisfy those constraints. Our preliminary evaluation on several case studies indicates that our approach is effective, and encourages further research in this direction. In particular, we plan to extend our framework to a variety of application domains, such as the Internet of Things, and to scale the model generation and reasoning modules to large distributed systems.

References

[1] "Owasp top 10 - 2017: The ten most critical web application security risks," The OWASP Foundation, Tech. Rep., 2017.

[2] M. Albanese and S. Jajodia, "A graphical model to assess the impact of multi-step attacks," *Journal of Defense Modeling and Simulation*, vol. 15, no. 1, pp. 79–93, January 2018.

Securing Distributed System Configuration through Optimization and Reasoning on Graphs

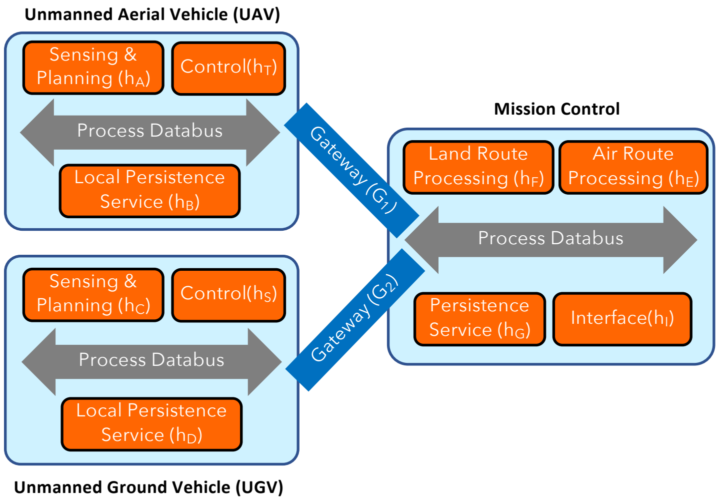
Hamed Soroush, Shantanu Rane and Massimiliano Albanese

1

ABSTRACT: Evaluating the security impact of configuration changes on a distributed system is an inherently complex challenge. Existing solutions simplify the problem by optimizing only individual components while ignoring complex interdependencies amongst components. In contrast, we construct a graph-based model of the system and its vulnerabilities that captures such dependencies. Inspired by a model that assesses the impact of multi-step attacks [1], we show how to reason about security impact of configurations. We employ SMT solvers to derive configurations that minimize the impact of attacks while preserving system functionality.

2

EXAMPLE SYSTEM:

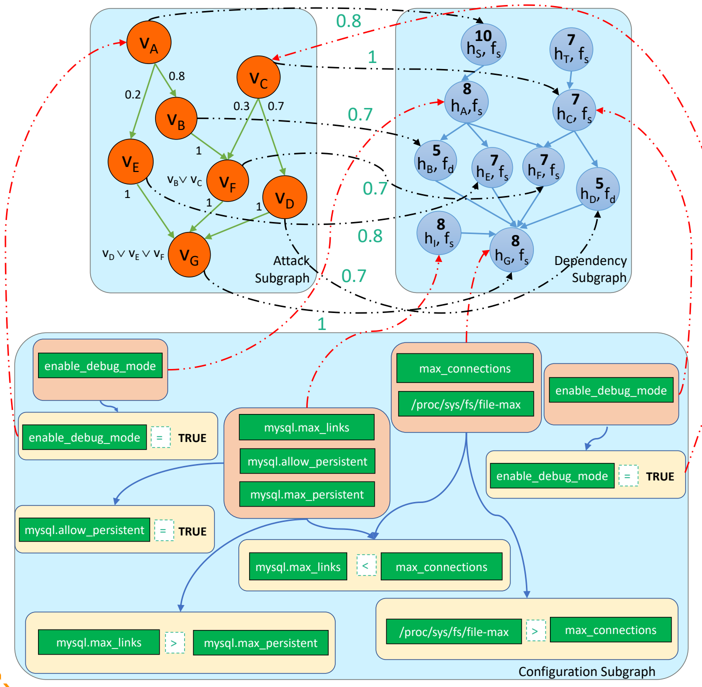

Unmanned Aerial Vehicle (UAV)
Sensing & Planning (h_A)
Control(h_T)
Process Databus
Local Persistence Service (h_B)
Gateway (G_1)
Mission Control
Land Route Processing (h_F)
Air Route Processing (h_E)
Process Databus
Persistence Service (h_G)
Interface(h_I)
Gateway (G_2)
Sensing & Planning (h_C)
Control(h_S)
Process Databus
Local Persistence Service (h_D)
Unmanned Ground Vehicle (UGV)

Globally optimal security decisions cannot be made without dependency information: dependencies help in analyzing the cascading impact of both attacks and configuration changes.

3

GRAPH-BASED MODELING APPROACH:

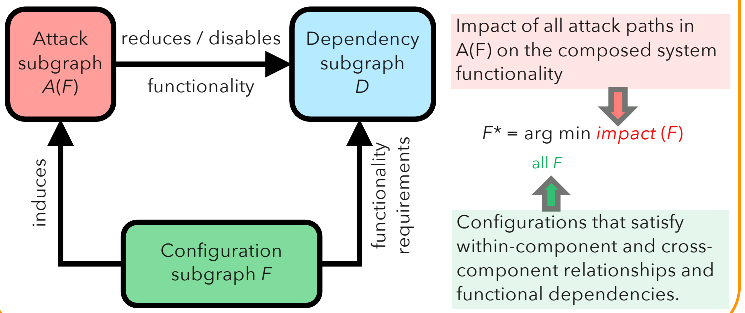
Express dependencies between configuration parameters, vulnerabilities, and functional components as a multi-layer graph.



4

PRELIMINARY RESULTS:

PRELIMINARY RESULTS: We built configuration, attack and dependency graphs for the above system in Neo4j, and solved the following optimization problem in Z3.


Attack subgraph A(F)
reduces / disables
functionality
Dependency subgraph D
induces
functionality requirements
Configuration subgraph F
Impact of all attack paths in A(F) on the composed system functionality
$F^* = \arg\min_{\text{all } F} \text{impact}(F)$
Configurations that satisfy within-component and cross-component relationships and functional dependencies.

5

ONGOING WORK:

- Automate the construction of configuration subgraphs from standard operating procedures and component specifications.
- Examine unsat core and tradeoff security against functionality.
- Provide evidence and explanation of secure configurations.

[1] M. Albanese and S. Jajodia, "A graphical model to assess the impact of multi-step attacks," *Journal of Defense Modeling and Simulation*, vol. 15, no. 1, pp. 79–93, January 2018.

This work is funded by the US Department of Defense under the DARPA ConSec program. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the DoD or any other agency of the U.S. Government.