Static Analysis Opportunities for Improving Agile and Moving Target Defenses

Anonymous Author(s)

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

agile security; moving target defense; access control; static analysis

1 INTRODUCTION

Agile defenses have been proposed to enable systems to change their defensive posture dynamically to thwart attacks. Researchers have suggested a variety of agile defenses that leverage renaming (e.g., for network services), migration (e.g., for cloud instances), variation (e.g., for application configurations), and patching (e.g., for programs), among others. These defenses appear promising to achieve the goal of agile defenses: to increase the adversary cost of launching a successful attack. However, agile defenses also incur a non-trivial cost to the defenders as well, leaving defenders hesitant to employ such defenses without further justification for their necessity and expense. A question we examine in this keynote is how to develop techniques that may aid defenders is choosing *when* to employ agile defenses and *which* agile defenses to employ.

Since agile defenses do incur a cost, one key question is when to employ agile defenses. When are the existing defenses like authentication and access control, which are inexpensive and/or broadly utilized already, insufficient to thwart attacks such that agile defenses should be applied to fulfill a gap in traditional defenses? Researchers have identified applications that may benefit from agile defenses, but what is it about these applications that make them better candidates (if they indeed are)? And, importantly, can we identify promising candidates automatically?

Once a candidate for an agile defense is identified, then another question is how to choose an agile defense to employ that may effectively. Of the various types of actions that can be taken dynamically to improve defenses, which should be chosen? How can we predict the impact of factors that may impact our decision among defenses? How does one balance trade-offs among multiple factors, such as performance and security, that impact the decision?

We have been examining answers to these questions as part of the Cyber Security Collaborative Research Alliance (Cyber-Security CRA). The Cyber-Security CRA is a consortium of academic, industry, government research laboratory researchers who are exploring the *science of cyber-decision making* to enable actors in military environments protect their resources from various threats. Our initial thoughts on agile security were presented in this workshop in 2014 [7]. Now, a little more than halfway through the project, we will discuss some areas of progress related to the above questions as well as research opportunities to explore moving forward.

Given the limited time, we will only be able to discuss one dimension of research on these questions, related to how *static analysis techniques* have helped us determine when and which agile defenses to employ. Static analysis techniques reason about models of system artifacts, e.g., networks, hosts, and programs, at rest. A wide variety of static analysis techniques have proposed and applied to a wide range of security problems. In this keynote, we will first examine how to think about the relationship between traditional and agile defenses to consider how static analyses may identify good candidates for agile defenses. Then, we will explore some ways that static analyses may be employed to triage systems, assess defenses, and apply agile defenses to worthy candidates.

2 EXPERIENCES WITH AGILE DEFENSES

We will discuss our experiences in applying static analyses to determine when to employ agile defenses and which agile defenses to employ. While we focus on static analyses, these analyses sometimes leverage evidence collected from program execution (e.g., logs) and/or dynamic analysis (e.g., fuzzing campaigns) identify where to perform static analysis.

First, we will discuss static analysis methods to improve software defenses. Our approach leverages static analysis methods to triage flaws found statically or dynamically (e.g., crashes) by searching for whether particular exploits are possible given adversary control over the flaw [5]. For flaws found to be at risk of exploitation, we have developed a separate static analysis to generate patches automatically that prevent exploitation systematically by ensuring that patches achieve *safety properties* [4].

Second, we will discuss static analysis methods to improve host defenses. Unlike the static analyses above for programs, these methods perform static analysis of access control policies that grant unsafe accesses [8]. Static analysis of access control policies enables us to determine which resources a program is authorized to access may be maliciously modified [10]. Using these results, we detect when programs access such resources to launch software analyses, such as the one above. Should triaging find a problem, we cann generate a "patch" of the access rules automatically [9].

Third, we will discuss static analyses to improve network defenses. We demonstrated that software-defined networks (SDNs) are prone to attacks that permit adversaries to reconstruct SDN policiesd [2]. Unlike host and software cases, we chose to develop a more systematic defensive approach that leverages multilevel security in network configuration [1]. However, SDN networks still see significant changes due to link failures, changes in traffic flows, etc., so we examine static analyses to maintain security using agile defenses.

Ultimately, we found that these diverse problems can be unified in single formalism enabling static analysis for configuring agile defenses system-wide. We will discuss modeling threats using an *attack graph* representation, which has long been used to compute attack paths in networks [3]. We will show how prior analyses

above can be modeled and composed as attack graphs to determine *when* to provide agility in defense.

To determine *which* agile action to take, we will discuss a technique to model the choice of defense as an integer programming problem, where we optimize one property (e.g., security) ensuring that all other properties (e.g., performance and functionality) satisfy constraints. We recently applied this approach to retrofit programs semi-automating using privilege separation [6].

3 AGILE DEFENSES AS DEFENSE IN DEPTH

We will close this keynote by exploring a bigger question of the relationship between agile and traditional defenses. We find that it is useful to think about agile defenses as another layer of defense in depth for your systems. However, traditional defenses create two types of risks that can be addressed by agile defenses: (1) reconnaissance risks and (2) attack surface risks.

First, weak traditional defenses grant adversaries wide latitude to perform reconnaissance to plan where and how to launch attacks. Agile defenses, particularly moving target defenses, have been applied to networks to thwart reconnaissance, but less so on programs and especially host access control. We explore how use of attack graphs may help determine how to make reconnaissance attacks prohibitively expensive.

Second, defenders currently do not track how weak traditional defenses introduces opportunities for attack, which we call a system's *attack surface*. A problem currently is that the defenders do not track the risks that they are taking systematically and often too many types of flaws are possible for each risk, so our defensive posture remains too ad hoc, often described as "penetrate-and-patch." We will explore how using attack graphs may guide decisions about how to employ agile defenses more effectively and more automatically.

4 CONCLUSIONS

In this talk, we discuss a variety of efforts in the Cyber-Security CRA project to leverage agile defenses using static analysis. We have explored static analyses for agile defenses applied to networks, hosts, and programs independently finding that despite the distinct domains there is much commonality. We describe how unify these problems in an attack graph representation to assess *when* to deploy an agile defense and describe our experiences in integer programming to choose among defenses. We close by examining how to apply agile defenses for limiting reconnaissance and attack surface risks using attack graphs to guide decision making.

REFERENCES

[1] Stefan Achleitner, Quinn Burke, Patrick McDaniel, Trent Jaeger, Thomas La Porta, and Srikanth Krishnamurthy. 2019. *MLSNet: A Policy Complying Multilevel Security Framework for Software Defined Networking.* Technical Report INSR-TR-500-2019. Institute of Networking and Security Research, Penn State University.

[2] Stefan Achleitner, Thomas La Porta, Trent Jaeger, and Patrick McDaniel. 2017. Adversarial Network Forensics in Software Defined Networking. In *Proceedings of the 2017 ACM Symposium on SDN Research.*

[3] Frank Capobianco, Rahul George, Kaiming Huang, Trent Jaeger, Mathias Payer, Srikanth Krishnamurthy, Zhiyun Qian, and Paul Yu. 2019. Employing Attack Graphs for Intrusion Detection. In *Proceedings of the 2019 New Security Paradigms Workshop (NSPW).*

[4] Zhen Huang, David Lie, Gang Tan, and Trent Jaeger. 2019. Using Safety Properties to Generate Vulnerability Patches. In *Proceedings of the 40th IEEE Symposium on Security and Privacy.*

[5] Kyriakos Ispoglou, Bader Al Bassam, Trent Jaeger, and Mathias Payer. 2018. Block Oriented Programming: Automating Data-Only Attacks. In *Proceedings of the 25th ACM Conference on Computer and Communications Security (ACM CCS).*

[6] Shen Liu, Dongrui Zeng, Yongzhe Huang, Frank Capobianco, Stephen McCamant, Trent Jaeger, and Gang Tan. 2019. Program-mandering: Quantitative Privilege Separation. In *Proceedings of the 26th ACM Conference on Computer and Communications Security (ACM CCS).*

[7] Patrick McDaniel, Trent Jaeger, Thomas La Porta, Nicolas Papernot, Robert J. Walls, Alexander Kott, Lisa Marvel, Anathram Swami, Prasant Mohapatra, Srikanth Krishnamurthy, and Iulian Neamtiu. 2014. Science and Security of Agility. In *Proceedings of the ACM Moving Target Defense Workshop.*

[8] Jonathon Tidswell and Trent Jaeger. 2000. An Access Control Model for Simplifying Constraint Expression. In *Proceedings of the ACM Conference on Computer and Communications Security.* 154–163.

[9] Hayawardh Vijayakumar, Xinyang Ge, Mathias Payer, and Trent Jaeger. 2014. JIGSAW: Protecting Resource Access by Inferring Programmer Expectations. In *Proceedings of the 23rd USENIX Security Symposium.*

[10] Hayawardh Vijayakumar, Joshua Schiffman, and Trent Jaeger. 2012. Integrity Walls: Finding attack surfaces from mandatory access control policies. In *7th ACM Symposium on Information, Computer, and Communications Security (ASIACCS).*