TRACTION: An Infrastructure for TRusted Alert sharing and Collaborative mitigaTION

Keywhan Chung[1], Phuong Cao[1], Yuming Wu[1], Zbigniew T. Kalbarczyk[1], Ravishankar K. Iyer[1], Alexander Withers[2]

[1] University of Illinois at Urbana-Champaign, [2] National Center for Supercomputing Applications
{kchung10,pcao3,ywu112,kalbarcz,rkiyer,alexw1}@illinois.edu

CCS Concepts

• **Security and privacy → Intrusion detection systems**; **Network security**; **Database and storage security**.

Keywords

Cyber Security, Intrusion detection, Security data sharing

1 Introduction

Advanced Persistent Threats (APTs) are among the most sophisticated attacks targeting networked systems. Instead of exploiting a single vulnerability, an APT uses multiple attack vectors to achieve objectives and may remain undetected for an extended period of time by staying under the radar of the defender's detection techniques. Such threats are not only growing in scale but coordinating to attack high-value sites, including both cyber and physical systems. As coordinated APTs are hard to detect with the limited data that can be collected from a single site, there is a need to enrich the observation of attacks by sharing information on monitored events with trusted sites. In this paper, we present our preliminary design of a new and unique shared infrastructure, *TRACTION* (Trusted Alert Sharing and Collaborative Mitigation), which at its core is a probabilistic graphical model, specifically, a distributed factor graph (DFG) anchored at each site by a local FG. The DFG provides an umbrella for automated and secure threat intelligence sharing. The overarching goal is to perform analysis and stop coordinated APTs in a manner previously not possible. Our initial design, at the scale of a single site, has been demonstrated in a production network at the National Center for Supercomputing Applications (NCSA) [1] at the Univ. of Illinois [3].

2 Threat intelligence sharing infrastructure

As depicted in Fig. 1, *TRACTION* considers a network of computing sites. Four trusted sites ($Site_1$, ..., $Site_4$) consist of multiuser, multinode, networked computer systems that provide computational services in which users remotely execute workloads on internal hosts (physical/virtual machines). Each site deploys a distinct set of trustworthy host- and network-based monitors (e.g., Zeek IDS) that capture attack activities. Sites output a common set of security alerts. Around the existing production and security monitoring system, *TRACTION* adds four core elements to each site (as depicted in Fig. 1.A): (i) a distributed database (❶ in Fig. 1), (ii) configurable honeypots (❷), (iii) an FG-driven detector (❸), and (iv) a distributed ledger (❹). To add onto the security logs generated from existing monitors, the honeypots attract attack traffic without impacting the production system. The rich data, and the intelligence derived from the FG-driven detector, are archived in the database. Based on predefined rules, the threat intelligence is shared across the sites in a dedicated data structure (see Fig. 1.B). In sharing the data, we deploy the concept of distributed ledger to validate the shared intelligence and to assure data confidentiality and integrity. In this poster, we focus on our DFG-driven detector.

3 Distributed factor graphs

Our approach is based on the theory of factor graphs that we use to fuse domain knowledge, experience from past incidents, and real-time observations from security monitors to detect and preempt ongoing attacks. We have demonstrated the effectiveness and timeliness of our FG-based detectors (e.g., AttackTagger [2]) using real data from NCSA's large-scale, high-performance computing infrastructure [3]. **Preliminaries.** Our overarching goal is to predict the state of a coordinated attack across sites. The state space for a user at a site can be represented as a set of 3 discrete states: $\Phi = \{benign, suspicious, malicious\}$. At any given time, a user state S on a site must be in one of the discrete states in Φ, which does not change during operation of the alert-sharing network. Security operators at each site define a set of discrete security alerts Ψ = { login, download, install modules , ... , logout }. An event E is a symbolic name that has a precise security semantics that is applicable to different operating systems and networking environments. **Local FGs at each site.** Given the random variables of alerts and attack stages, we define a probability distribution $P(E_1, E_2, \ldots, E_n, S_1, S_2, \ldots, S_n)$ over the random variables. This probability distribution represents a *FG*, a factorization of that describes the statistical relationships between the observed alerts and the states. In our local FG, we model this probability distribution as a chain-structured bipartite graph that captures the sequence of event/state variables at each time point. Since the structure of the probability distribution is known, it can be simplified to a product form $P(E_1, E_2, ..., E_n, S_1, S_2, ..., S_n) = \Pi f_c(x_c)/Z$, in which x_c is a set of random variables (e.g., a state or an event), f_c is a factor function, and Z is a normalization factor to ensure a proper probability distribution. The local factor graph estimates the most likely sequence of states, S, that maximizes the value of the probability distribution function for a given sequence of events E. **Distributed FG across sites.** We extend the notion of the local FG to a network of distributed FGs that communicates the messages indicating ongoing attacks across a set of trusted sites (Υ). In the following,

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).
HotSoS, April 1–3, 2019, Nashville, TN, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7147-6/19/04.
https://doi.org/10.1145/3314058.3317292

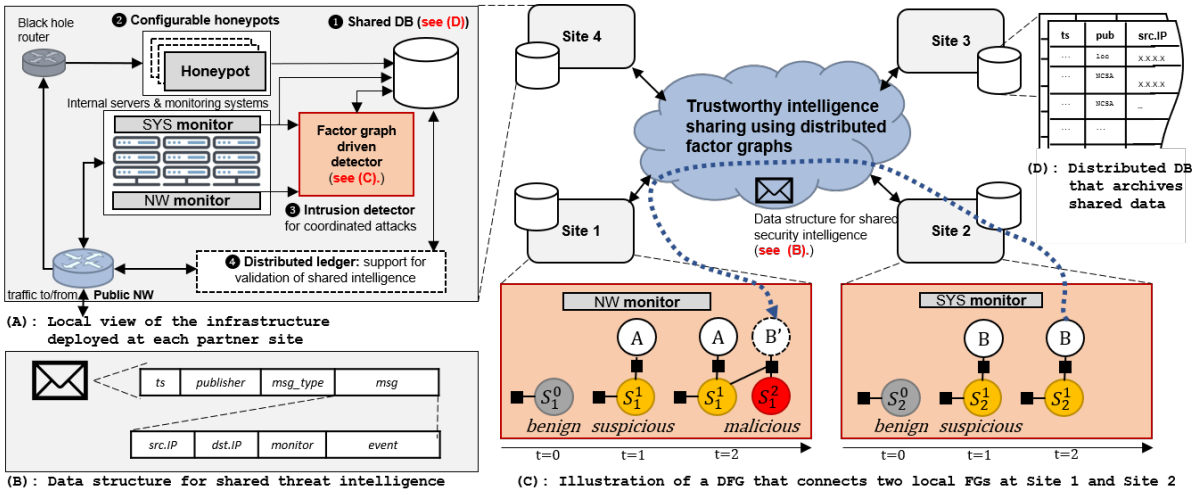


Figure 1: Overview of the design of the TRACTION infrastructure.

we briefly describe the mathematical model that underlies the design and implementation of the DFG and its integration into the overall TRACTION system. Some of the design issues include (i) reliability of links, (ii) message (i.e., shared security data) ordering, (iii) management of asymmetric levels of trust across partnering sites, and (iv) adaptive definition of thresholds to explicitly determine the threat intelligence to be shared. These issues have been extensively studied in the literature, and the challenge here is to integrate the solutions into the design. **Model.** A distributed factor graph $DFG :=< FG_{ij}, g_{ij} | i, j \in \Upsilon >$ consists of local factor graphs at each site FG_i. When a $Site_i$ notifies another $Site_j$ of an event, an intermediate factor function g_{ij} connects the two sites and forms a DFG. **Pre-sharing agreements**. For an event to be shared between two sites, the event must be comprehensible by both sites. A shareable event must be in the intersection of the two alert sets in two sites. Formally, $share : \Upsilon \times \Upsilon \times \Psi \rightarrow \{0, 1\}$ is a set membership function that determines whether a $Site_i$ would share the event E with a $Site_j$. **What to share and when to share.** Upon observing a new event, a site reevaluates its internal state and composes a message to be shared with neighbor sites via a secure communication channel. A message $m_{i \rightarrow j}$ from a $Site_i$ to another $Site_j$ is composed of two components: (i) a security event E_n observed at $Site_i$ at a time n and (ii) a belief b that an attack is occurring, which is a 3×1 vector indicating the value of the probability mass function on each of the states in Φ, i.e., the belief. **Post-sharing update of states.** When a message $m_{i \rightarrow j}$ is shared from $Site_i$ (S_i) to $Site_j$ (S_j), the S_j increases its time step $t \rightarrow t + 1$. The S_j does not need to include all past alerts and factor functions of the sending site, S_i. The receiving site (S_j) processes each component of the message as follows. For the shared security event, S_j runs a Boolean acceptance function $Accept : \Psi \rightarrow \{0, 1\}$ to include only significant alerts in its graph, using its learned knowledge base of factor functions. If the new event is accepted, S_j processes it as the local factor graph. For the belief (b) that an attack is occurring, S_j uses the intermediate factor function g_{ij} to incorporate the belief into its local FG, then performs inference.

We illustrate the evolution of a DFG in Fig. 1.C. An $Site_i$ creates a new random variable S_i^{t+1} to reflect the site's (unknown) state at a time t and treats the belief as new information. The site incorporates that new information into its existing local FG by attaching a factor function $g_{i,j}$ to the belief and the new state S_i^{t+1}, and reevaluates the factor graph to determine the state at $t + 1$. These factor functions ($g_{i,j}$) are domain-specific and must be predefined.

t	Site 1		Site 2	
	FG evaluation	State	FG evaluation	State
0	$S_1 = argmax p_1(S_1)$ $S_1 = [\mathbf{0.9}, 0.05, 0.05]$	benign	$S_2 = argmax p_2(S_2)$ $S_2 = [\mathbf{0.98}, 0.01, 0.01]$	benign
1	$S_1 = argmax p_1(S_1) f_1(S_1, A)$ $S_1 = [0.4, \mathbf{0.5}, 0.1]$	susp.	$S_2 = argmax p_2(S_2) f_2(S_2, B)$ $S_2 = [0.3, \mathbf{0.5}, 0.2]$	susp.
2	$S_1 = argmax$ $p_1(S_1) f_1(S_1, A) g_{12}(S_1, B')$ $S_1 = [0.1, 0.3, \mathbf{0.6}]$	**mal.**	$S_2 = argmax p_2(S_2) f_2(S_2, B)$ $S_2 = [0.4, \mathbf{0.5}, 0.1]$	susp.

Table 1: State transition of a set in three time steps following our motivating example in Fig. 1.C.

4 Future Work

In designing *TRACTION*, we have used live traffic at NCSA to test the prototypes of individual components that will constitute the cornerstones of our design. The next step is to define an interface logic to allow for proper information flow among the components and across the sites.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 15-35070. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Alexander, Withers. 2017. Science DMZ Actionable Intelligence Appliance (SDAIA). https://wiki.ncsa.illinois.edu/display/cybersec/SDAIA

[2] Phuong Cao, Eric Badger, Zbigniew Kalbarczyk, Ravishankar Iyer, and Adam Slagell. 2015. Preemptive Intrusion Detection: Theoretical Framework and Real-world Measurements. In *Proc. of the Symp. and Bootcamp on the Science of Security*. 5:1–5:12.

[3] Phuong M. Cao, Yuming Wu, Subho S. Banerjee, Justin Azoff, Alex Withers, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2019. CAUDIT: Continuous Auditing of SSH Servers to Mitigate Brute-Force Attacks. In *Proc. Symp. on Networked Systems Design and Implementation*. https://www.usenix.org/conference/nsdi19/presentation/cao