Towards an Unified Security Testbed and Security Analytics Framework

Phuong Cao, Eric C. Badger, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer
Coordinated Science Laboratory
University of Illinois at Urbana Champaign
{pcao3,badger1,kalbarcz,iyer}@illinois.edu

Alexander Withers, Adam J. Slagell
National Center for Supercomputing Applications
University of Illinois at Urbana Champaign
{alexw1,slagell}@illinois.edu

ABSTRACT

This paper presents the architecture of an end-to-end security testbed and security analytics framework, which aims to: i) understand real-world exploitation of known security vulnerabilities and ii) preemptively detect multi-stage attacks, i.e., before the system misuse. With the increasing number of security vulnerabilities, it is necessary for security researchers and practitioners to understand: i) system and network behaviors under attacks and ii) potential effects of attacks to the target infrastructure. To safely emulate and instrument exploits of known vulnerabilities, we use virtualization techniques to isolate attacks in containers, e.g., Linux-based containers or Virtual Machines, and to deploy monitors, e.g., kernel probes or network packet captures, across a system and network stack. To infer the evolution of attack stages from monitoring data, we use a probabilistic graphical model, namely AttackTagger, that represents learned knowledge of simulated attacks in our security testbed and real-world attacks. Experiments are being run on a real-world deployment of the framework at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.

1. SYSTEM ARCHITECTURE

In this section, we describe components of our system including the monitoring infrastructure, log processing, log analytics, and security testbed (Figure 1).

Monitoring architecture. The National Center for Supercomputing Applications (NCSA) provides raw log data including syslog, netflow, and Bro Intrusion Detection System (IDS) [1] logs gathered from over 5000 nodes and high speed networks. The logs also include activity and data centered around BlueWaters and the Extreme Science and Engineering Discovery Environment (XSEDE) at the University of Illinois at Urbana-Champaign, both of which cater to thousands of users.

NCSA's *syslog* data is aggregated from both user systems and servers into a redundant, fault tolerant rsyslog setup. Real-time streams of syslog data, mainly focusing on authentication and application logs are sent. The NCSA maintains a Bro IDS cluster to monitor incoming and outgoing traffics of the NCSA facility and BlueWaters. The *Bro IDS logs* are collected by the cluster, aggregated into hourly files, and read by a Splunk forwarding agent. To help augment the syslog and IDS logs, *netflow* logs provide basic network traffic information. The NCSA collects all netflow data at a single point using nfdump and forwards a copy of netflow data using Samplicator.

The collected syslog, Bro IDS logs, and netflows are collected at NCSA using rsyslog and forwarded to the Log Pre-Processing module using rsyslog's routing capabilities.

In addition to NCSA's log data as described before, the NCSA's Incident Response Team (IRT) provides data on security incidents that occur with the NCSA and affiliated organizations for which the IRT is responsible. The data includes incident reports and log data that is normally collected by NCSA's log aggregation systems. Furthermore, security breach data is provided using HoneyPot and HoneyNet setups at the NCSA. Such setups gives a baseline set of data that can be used to test our AttackTagger preemptive intrusion detection framework against the real-world attacks [2].

Log processing. The aggregated logs need to be pre-processed in order to minimize the amount of information to analyze. The log streams forward their data to the log pre-processor, which: i) picks out only the pieces of data that are meaningful to our system and ii) puts them into a consistent format that is consumable by AttackTagger. For example, relating to: i), if we want to only care about authentication protocols, then we can filter out everything except autentication logs, such as ssh logins. Relating to ii), AttackTagger needs the logs to be put into a format containing 4 fields: "timestamp", "event", "user", and "description" so that it can perform the uniform operations on all of the different log types. The pre-processor then forwards the output to Kafka so that it can queue the messages for consumption by AttackTagger.

Log transfering. Kafka is a high-throughput distributed messaging system [4]. Kafka comes with many attractive features for our use. First of all, Kafka is high-throughput and horizontally scalable. It is able to handle hundreds of megabytes of reads and writes per second from thousands of clients. In addition, Kafka offers strong durability and fault-tolerance guarantees. Because the system was designed to

HotSoS '15 Urbana, IL, USA

http://dx.doi.org/10.1145/2746194.2746218 ...$15.00.

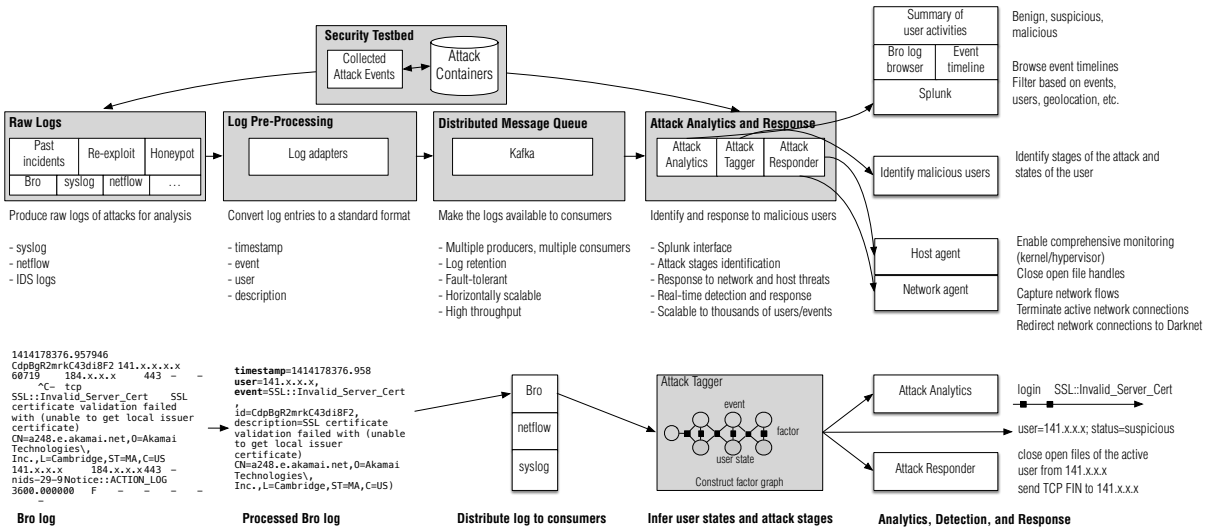

Figure 1: System architecture of security testbed and analytics of security logs. Processing and analytics of an example Bro log entry is illustrated.

be distributed, it is seamless to add new machines without downtime and increase the capabilities of the system. We use the Kafka API to set up our log pre-processor as a producer (or multiple producers). Kafka then delivers these messages to AttackTagger, which. Since Kafka has the ability to have multiple producers and consumers, we have the option to choose to either set up a consumer/producer pair for each log type or aggregate all of them together into a single stream. Currently, all of these log types are in different streams. Kafka is also able to store the logs for a configurable amount of time so that we can roll-back through old logs. We have Kafka set up to evict logs after 7 days.

Security testbed. A security testbed is a controlled system and network environment where attacks can be replicated. Main requirements of a security testbed are: i) isolated to prevent an attack affecting production infrastructure, ii) instrumented to collect system and network events in various attack stages, and iii) repeatable and convenient to share among security researchers.

We propose a security testbed architecture based on two virtualization technologies: Virtual Machine Monitors (VMM) at the hardware-level and Linux Containers (LXC) at the operating-system-level to emulate multiple isolated systems on a single control host [3, 5].

Isolation. While LXCs create an isolated system using process's models on a shared kernel, VMMs create an isolated system by spawning a full-featured virtual machine. Compared to VMMs, LXCs are more lightweight and can only be used to replicate application-level attacks, e.g., an SQL injection attack in a web applications or a buffer overflow vulnerability. VMMs can replicate kernel-level attacks, e.g., an integer overflow in a device driver.

Instrumentation. We deploy various monitors across the system stack and network to collect events during the attacks. We use kernel probes to collect kernel events, syslog to collect application events, and netflow to collect network events. Most of the monitors are deployed on the same system where we run our attacks. Advanced monitors, such as HyperTap support continous, event-driven VM monitoring, which enables out-of-VM capturing the host system state [6]. Deploying monitors in separated containers or VMs reduces attack surface of contaminating the monitors.

Repeatability. Each attack is packaged using a group of containers or VMs that contain the system that the attack is run on and the corresponding monitors. We organize attacks into a centralized registry, where a security researcher can clone the attacks into their local machine for educational purposes.

Monitoring data from the security testbed is constantly delivered to our probabilistic graphical model, i.e., AttackTagger, as feedback to improve attack detection.

2. CONCLUSION

In this paper, we described the architecture of an end-to-end security testbed and security analytics framework. Our security testbed has following benefits: i) facilitating sharing of security threats among security analysts, and ii) serving as an attack database for convenience of understanding attacks. Our preemptive intrusion detection framework provided a new application of probabilistic graphical models to identify progressing attack at early stages.

3. REFERENCES

[1] Bro, I. Homepage: http://www. bro-ids. org, 2008.

[2] Cao, P., Badger, E., Kalbarczyk, Z., Iyer, R., and Slagell, A. J. Preemptive intrusion detection: Theoretical framework and real-world measurements. In *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security* (2015), HotSoS '15, ACM.

[3] Goldberg, R. P. Survey of virtual machine research. *Computer 7*, 6 (1974), 34–45.

[4] Kreps, J. e. a. Kafka: a distributed messaging system for log processing. In *6th International Workshop on Networking Meets Databases* (2011).

[5] Merkel, D. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal 2014*, 239 (2014), 2.

[6] Pham, C., Estrada, Z., Cao, P., Kalbarczyk, Z., and Iyer, R. K. Reliability and security monitoring of virtual machines using hardware architectural invariants. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on* (2014), IEEE, pp. 13–24.