Poster: CompareView - A Provenance Verification Framework for Detecting Rootkit-Based Malware

Chehai Wu (Student)
Department of Computer Science
Rutgers University, New Brunswick
wuc@cs.rutgers.edu

Danfeng (Daphne) Yao (Faculty)
Department of Computer Science
Rutgers University, New Brunswick
denfeng@cs.rutgers.edu

Abstract

Using rootkit mechanisms to hide malware presence is pervasive in today's computer attacks. We propose the CompareView *framework, a host-based solution to detect stealthy outbound traffic generated by rootkit-based malware. Using a lightweight cryptographic protocol, our CompareView framework compares the views of outbound network packets at different layers of the host network stack and verify the* provenance *information of each packet. CompareView identifies and blocks suspicious network traffic that is not accompanied with proper digital signature stating its origin. The prototype is implemented in Windows XP and leverages on-chip TPM to enforce its own integrity. Our evaluation results show that our provenance verification approach is effective and efficient.*

1. Introduction

Rootkit is a mechanism that hides malware from detection. Malware equipped with rootkit is extremely difficult to detect. Existing work has been mainly focused on operating system level detection, including identifying suspicious system call execution patterns[1], discovering vulnerable kernel hooks[5], or using virtual machine to enforce correct system behaviors[2][3]. Existing OS level detection methods are quite effective, but typically require sophisticated and complex examination of kernel instruction executions. To enforce the integrity of the detection systems, virtual machine monitor (VMM) is usually required such as in [3]. Although powerful, widely deploying VMMs for protecting average user' PCs has not yet become a reality.

Most malware constantly communicates with the outside world, for the purpose of exporting sensitive data. Our detection is based on the observation that there are intrinsic differences between how a person and malware interacts with a computer. Legitimate outbound network traffic initiated by humans passes through the entire network stack in the host's operating system. In comparison, rootkit-based malware typically bypasses higher layer inspections in the network stack by directly calling lower-level network functions.

We explore the network stack and packet properties of outgoing traffic generated by humans and malware, and develop a robust cryptographic protocol for enforcing the proper *packet provenance on the network stack*. We describe the design and implementation of our *CompareView* framework. The goal of CompareView is to identify and block suspicious network activities that belong to malware. Our framework defends against rootkit-based malware by comparing the views of outgoing network packets at different layers of the network stack and verifying packet provenance information. We also utilize on-chip Trusted Platform Module (TPM), which is available for most commodity computers, for key generation and integrity protection.

2 Overview of CompareView

We assume a powerful type of malware that sends outbound traffic and is capable of hiding its presence in the user space of the operating system.

Architecture. CompareView is an add-on to the host's network stack. It consists of a Sign Module and a Verify Module shown in Figure 1. Sign Module is at the upper edge of the transport layer while Verify Module is at the lower edge of the network layer. All legitimate network packets pass through Sign Module first and then Verify Module. Sign Module signs every packet and sends signatures as packet provenance information to Verify Module which verifies them. If packets' signatures cannot be verified, then they should be labeled suspicious as they bypass Sign Module and are likely generated by stealthy malware.

Key Management and System Integrity. When the system starts up, Sign Module and Verify Module generate their public/private key pairs and notify public keys to each other. Using the public/private key scheme, the two modules securely exchange two symmetric keys. One is for signature generation and verification; the other is used to secure signature transfer from Sign Module to Verify Module. To ensure that the integrity of CompareView framework

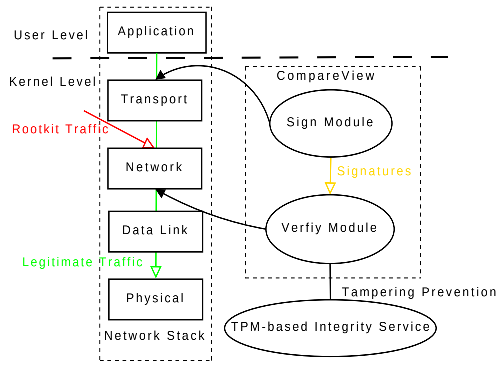

Figure 1. Schematic drawing of components in CompareView and their interactions with the host's network stack. Legitimate traffic origins from application layer whereas rootkit traffic is injected into the lower layers.

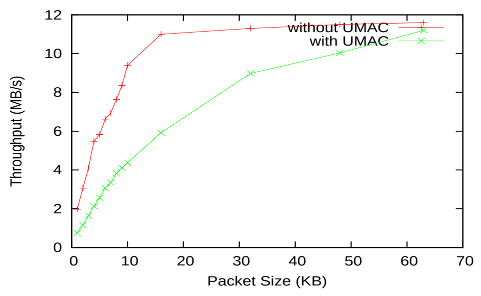

Figure 2. Performance comparison with or without CompareView.

and signing key secrecy, CompareView leverages on-chip TPM to derive signing keys and to attest kernel integrity at boot along with CompareView modules, details of which are omitted due to space limit.

Prototype Implementation. We implement CompareView in Windows XP. Sign Module is realized as a TDI filter device at the upper edge of the transport layer in the Windows TCP/IP stack. All legitimate network packets from Winsock APIs is captured and signed by Sign Module. Verify Module is an NDIS intermediate miniport driver at the lower edge of the network layer. It intercepts and verifies all packets just before they are sent to network interface card drivers. The signature algorithm is UMAC which is fast and lightweight. Intuitively, Verify Module at the network layer has to reassemble ethernet frames in order to reconstruct the original transport layer data segments and then compute signatures. However, because UMAC computes signatures incrementally and outgoing ethernet frames in the network stack are in order, Verify Module does not need to reassemble fragments. It updates the corresponding signature for each fragment on-the-fly, which significantly reduces the time and memory costs. It is important to note that packet signature is not appended to each packet. Otherwise the removal of signature of a packet in Verify Module would lead to recalculation of checksums, which is inefficient. To solve this problem, in CompareView Sign Module sends encrypted signatures directly to Verify Module as shown in Figure 1. Signatures are kept in a hash table indexed by packet source and destination addresses and ports for fast lookup.

3. Experimental Evaluation

We first test against a piece of proof-of-concept malware that can bypass the transport layer to send outgoing packets. Our experiments show that CompareView's Verify Module detects such an attack. However, the malware can disable URL filtering functionality of Trend Micro OfficeScan Client. An extended version of CompareView implementation is able to identify real-world rootkits (weaker than our proof-of-concept malware), including `Fu_Rootkit`, `hxdef`, and `AFXRootkit`, all of which hide process information and opening ports.

Figure 2 shows the network throughput with and without using CompareView. With CompareView, the throughput is lower in general. Yet, as the size of packet grows, the throughput is close to the ideal value. The observed performance degradation is minimal and acceptable in practice, because typically PCs have low upstream traffic even with P2P applications running.

4. Work-In-Progress

Our CompareView framework provides a very flexible and trustworthy setup for performing advanced host-based traffic inspection. We are currently working on a complex input-traffic correlation analysis method for malware traffic detection. We trace the user inputs in their online activities and match the events with outgoing traffic. We use a trusted computing platform as in TUBA [4] to prevent the injection of fake user input events. We will use advanced certification and visual analytic techniques to distinguish third-party contents in web from malware traffic.

References

[1] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *ESEC-FSE*, 2007.

[2] A. Dinaburg, P. Royal, M. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *CCS*, 2008.

[3] A. Srivastava and J. Giffin. Tamper-resistant, application-aware blocking of malicious network connections. In *RAID*, 2008.

[4] D. Stefan and D. Yao. Keystroke dynamics authentication and human-behavior driven bot detection. Technical report, Rutgers University, 2008.

[5] Z. Wang, X. Jiang, W. Cui, and X. Wang. Countering persistent kernel rootkits through systematic hook discovery. In *RAID*, 2008.