Towards A Usable Provenance Reference Monitor

Adam Bates, Kevin Butler

Computer and Information Science Department, University of Oregon

What is Provenance?

The provenance of an object is a detailed history describing the entities and processes involved in producing, delivering, and storing that object. Provenance-aware systems gather and report metadata that describes the history of each object (e.g., files, network messages) being processed. As the world becomes increasingly distributed and reliant on cloud computing, there is a growing interest in developing such systems, as they allow users to track, and understand, how a piece of data came to exist in its current state on the system. *Unfortunately, the provenance-aware systems that exist today are insecure, operating under very different models and assumptions, pointing to a pressing need for a dedicated platform for provenance development.*

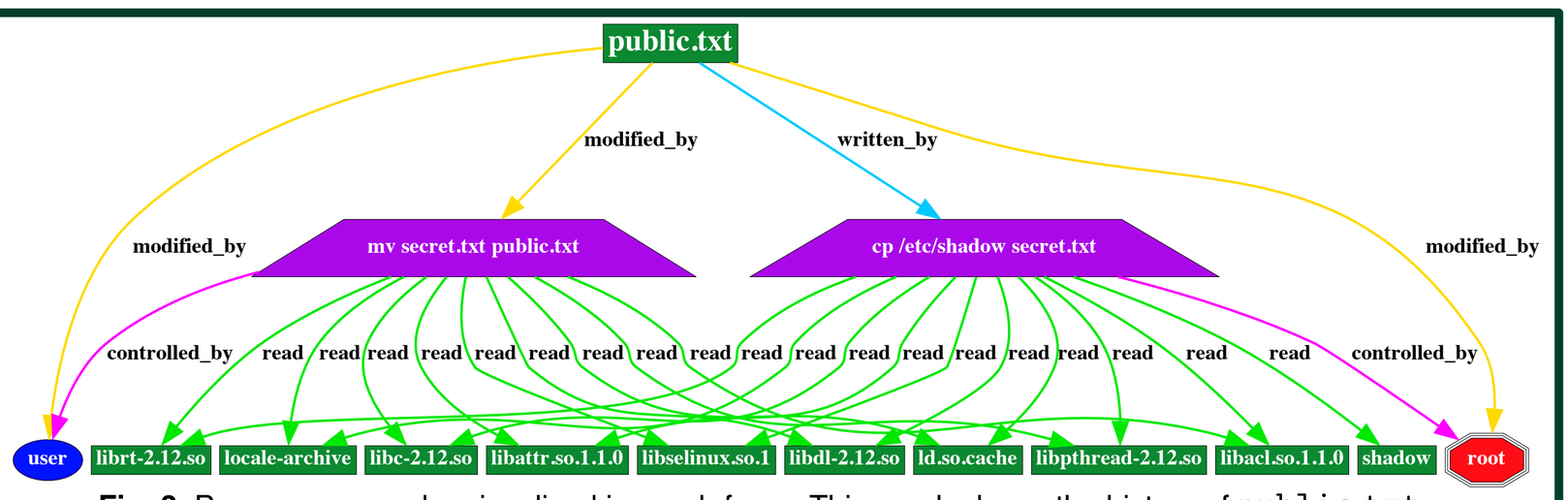

Fig. 3. Provenance can be visualized in graph form. This graph shows the history of `public.txt`.

Secure Provenance

The security of provenance data is critical, but actually requires different protections than the data that it describes. A data object may be public, but its provenance may leak sensitive info about the process through which it was derived (e.g., lab tests). It is equally important that provenance be protected from attack, leading some to call for *provenance reference monitors*. A reference monitor is a small mechanism that enforces security for an entire system, such as SELinux or Windows SRM. In the case of provenance reference monitors, the mechanism must be able to collect provenance on all system events (**complete observation**) in a manner that cannot be avoided or subverted by an attacker (**tamperproofness**). When provenance security is assured, it can be used in many applications. Figure 1 shows how provenance can assess integrity. Figure 2 shows how it can provide evidence of an unauthorized system access.

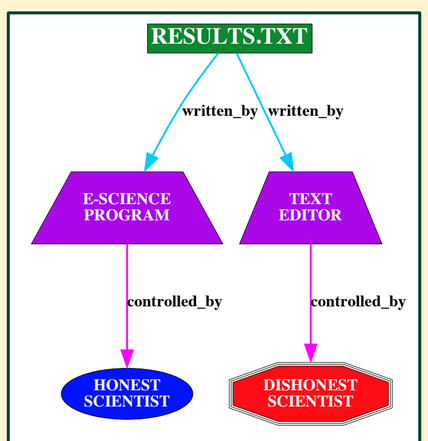

Fig. 1. Provenance can assess data integrity. Here, we see that `results.txt` has been manually edited.

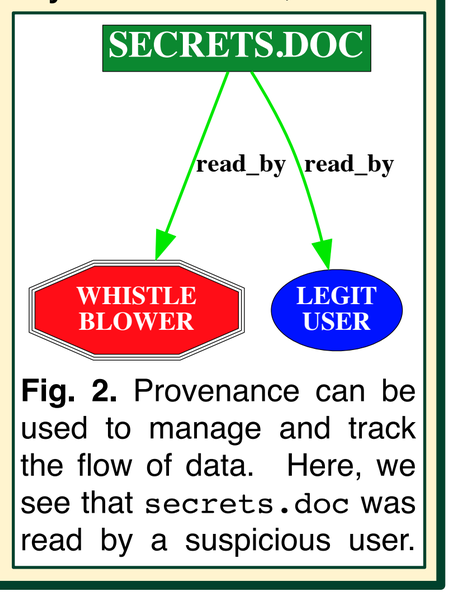

Fig. 2. Provenance can be used to manage and track the flow of data. Here, we see that `secrets.doc` was read by a suspicious user.

Linux Provenance Modules

We present the Linux Provenance Module Framework (LPM), enabling the development of provenance monitors in the Linux operating system. Figure 4 shows that LPM forms a provenance layer that observes all activity from within the Linux kernel. LPM does not interfere with security; thus, LPM can be protected by Linux's existing security mechanisms, which is both easier and safer. We wrote an SELinux policy to protect LPM's trusted computing base.

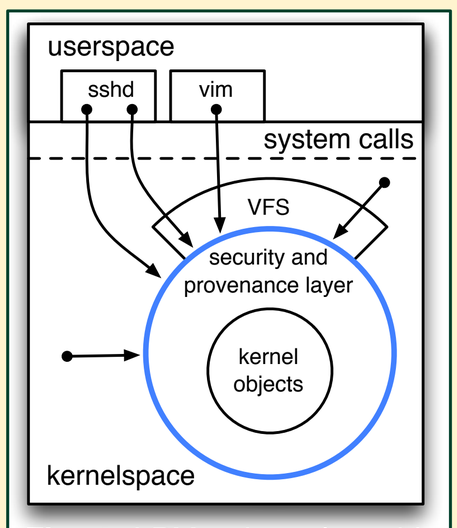

Fig. 4. LPM exists alongside Linux's security framework.

Proposal	Layer	Application Context?	File System?	IPC?	Memory?	Network?	Processes?
HI-FI	Kernel (LSM)		✓	✓	✓	✓	✓
Lineage	Kernel		✓	✓		✓	✓
PASS	Kernel (VFS)	Optional	✓	✓		✓	✓
QUIRE	Platform			✓			
REDUX	Application	✓					
SNooPy	Application					✓	
SProv	Application		✓				
Trio	Application	✓					

Fig. 5. Past proposals for automatic provenance collection vary by scope and operational layer.

Past Proposals. LPM is designed to be a general platform for provenance collection, so we started by considering the needs of past provenance-aware systems. Figure 2 shows that these systems vary in the events for which they collect provenance, such as application context, files, inter-process communication (IPC), memory, network events, and process executions.

Provenance Hooks. LPM is able to serve all of these needs, observing system activity through a set of 170 *provenance hooks* that are placed throughout the kernel. We have placed a provenance hook directly after each security hook in the kernel, facilitating provenance collection for all activities permitted by the active security policy. LPM also lets applications annotate provenance for events that cannot be viewed from within the kernel, such as workflows or database queries.

High-Fidelity Provenance

The first provenance monitor for LPM is a re-implementation of the Hi-Fi system. Hi-Fi collects provenance over all system activity (see Figure 5) while imposing just 3% performance overhead. Hi-Fi's provenance can be used to offer a detailed history of any system object. Figure 3 shows the history of a seemingly innocuous file, public.txt. Should a non-privileged user be allowed to read this file? It turns out the answer is *no*; public.txt actually leaks data from the `/etc/shadow` file, where Linux stores its passwords. By checking the provenance, we may have prevented a hacker from stealing our passwords from this machine.

Policy-Reduced Provenance

In practice, provenance monitors record extraneous information, such as the provenance for starting up the system, creating excessive storage overhead. We are developing a new module that can selectively collect provenance based on a user-specified policy. A key insight is that our policy can leverage existing context from a system's security framework. We will be using this module to discover new *attack surfaces* that exist in popular programs like Firefox and Dropbox, improving application security.

Conclusion

The LPM Framework will bring usable, secure provenance monitors to the Linux operating system. We will be releasing our source code upon publication, and intend to pursue incorporating LPM into the mainline Linux kernel source tree.

OSIRIS Oregon Systems Infrastructure Research & Information Security Laboratory

This research is supported by NSF Grant CNS-1254198 and by Massachusetts Institute of Technology Lincoln Laboratory.

For further information, please contact
Adam Bates < *amb@cs.uoregon.edu* >,
Kevin Butler <butler@cs.uoregon.ed>