

LLSS: Toward System Support for Plugging Privacy Leaks

Aydan R. Yumerefendi and Landon P. Cox
Duke University, Durham, NC
{aydan, lpcox}@cs.duke.edu

Imagine that two users, Alice and Bob, have a conversation about a sensitive subject over email. Alice is vigilant about protecting the secrecy of the discussion and encrypts all copies of her sent and received emails. Bob, on the other hand, is well-meaning but negligent, and does not bother to encrypt any of his on-disk files. Worse, he unwittingly stores his email files in the portion of his hard drive that is shared with the rest of the Internet via a peer-to-peer client. One night, unbeknownst to either Alice or Bob, someone on the peer-to-peer network peruses Bob's shared space, finds his email files, and reads their private conversation. Despite Alice's best efforts and Bob's best intentions, the privacy of their discussion has leaked.

While only hypothetical, this scenario is frighteningly plausible. A 2003 usability study of the Kazaa peer-to-peer file-sharing network [1] found that many users share their entire hard drive, including email inboxes and credit card information. Over 12 hours, the study found 156 distinct users who were sharing their email inboxes. Not only were these files available, but other users were observed downloading them.

In both the hypothetical and empirical example, the users were not malicious, nor had the programs running on their behalf been compromised. Yet sensitive information still leaked onto the Internet because many users do not understand how software components interact.

It is easy to see why. Ostensibly, a Kazaa client and email program have no relationship. They are from different vendors, are configured separately, and, in general, do not use any of the same files. This is by no means an isolated example. Experts [2, 3] and non-experts alike struggle to protect their sensitive information. Of course, any solution to this problem must be practical. Ideally, it would impose little administrative overhead, support legacy code, and avoid large performance penalties.

To this end, we propose a new privacy management system called *LLSS*¹. LLSS makes privacy management simpler by helping users define *what* data is important and *who*

is trusted, rather than forcing them to understand the complex dynamics of *how* data flows. Our approach can be decomposed into two complementary goals—first, to create file and host meta-data, such as labels and group membership, and second, to use this meta-data to alert users about potential breaches.

As in other privacy management systems, labeling sensitive files is crucial. Because many users may find this process prohibitively difficult or inconvenient, LLSS identifies private information by searching for files that contain data such as email, credit card numbers, and Social Security numbers. Once labeled, sensitive information should only be exchanged within a mutually trusting group. To help users define who they trust with what data, LLSS questions users about hosts as suspicious data flow arises.

Complementary to the issue of generating file and host meta-data is how it can be used to prevent leaks. One promising approach is to use meta-data and speculative execution to infer in-flow and out-flow dependencies. For example, if process P tries to read from sensitive file, F, LLSS returns the content of F to P, but returns an equal amount of garbled data to a speculative copy of P, denoted SP. LLSS then runs P and SP in parallel, delivering any events or network requests intended for P to SP as well.


If P subsequently attempts to write to a socket that is bound to an untrusted host, LLSS examines the content of the buffer and waits a short amount of time for SP to do the same. If SP attempts to write the same data to the same host, then P's write is allowed, since the output cannot depend on the content of F. However, if SP attempts to write different data than P, P might be leaking sensitive information to an untrusted host and the user is asked to adjudicate the write.

Through these and other techniques, LLSS aims to provide practical privacy management by helping users define system meta-data while unburdening them of the need to comprehend complex system dynamics.

1. REFERENCES

- [1] N.S. Good and A. Krekelberg. Usability and privacy: a study of KaZaA P2P file-sharing. In *Proceedings of SIGCHI*, pages 137–144, Ft. Lauderdale, FL, April 2003.
- [2] Associated Press. Miami university warns students of privacy breach. *Akron Beacon Journal*, September 16, 2005.
- [3] L. Vaas. Hack at UC Berkeley potentially nets 1.4 million SSNs. *EWeek*, October 20, 2004.

¹LLSS stands for “Loose Lips Sink Ships”


 Duke Systems

LLSS: Toward System Support for Plugging Privacy Leaks

Aydan R. Yumerefendi and Landon P. Cox
 Duke University
 {aydan, lpcox}@cs.duke.edu

Privacy leak scenario




Alice





Bob




Despite Alice's best efforts, Bob's best intentions, and no system compromises ...


 Duke Systems

Aydan R. Yumerefendi and Landon P. Cox
 LLSS: Toward System Support for Plugging Privacy Leaks
 SIGSP '09: Work-in-Progress Session

Frighteningly plausible


- “Usability and privacy: a study of Kazaa...”
 - Good and Krekelberg, SigCHI 2003
 - In 12 hours, found 150 inboxes on Kazaa
 - Observed people downloading them
- Problem
 - People don't understand software interactions
- LLSS [“Loose Lips Sink Ships”]


 Duke Systems

Aydan R. Yumerefendi and Landon P. Cox
 LLSS: Toward System Support for Plugging Privacy Leaks
 SIGSP '09: Work-in-Progress Session

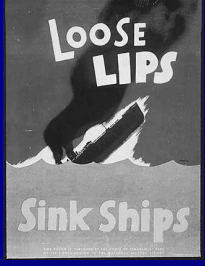
LLSS approach


- Ask users **What** and **Who**, not **How**
- Help users define system meta-data
 - Find sensitive files (email, ssn, cc numbers)
 - Infer other trusted hosts
- Use meta-data to identify potential leaks
 - Interpose on file system and socket syscalls
 - Use speculative execution to detect leaks


 Duke Systems

Aydan R. Yumerefendi and Landon P. Cox
 LLSS: Toward System Support for Plugging Privacy Leaks
 SIGSP '09: Work-in-Progress Session

Questions?




 Duke Systems

Aydan R. Yumerefendi and Landon P. Cox
 LLSS: Toward System Support for Plugging Privacy Leaks
 SIGSP '09: Work-in-Progress Session

Figure 1: Work-in-Progress Session Slides