Packets with Provenance[2]

Anirudh Ramachandran, Kaushik Bhandankar, Mukarram Bin Tariq, and Nick Feamster
School of Computer Science, Georgia Institute of Technology
{avr,kaushikb,mtariq,feamster}@cc.gatech.edu

ABSTRACT

Traffic classification in today's networks relies on coarse-grained or ephemeral fields of network packets, such as IP addresses or port numbers, which often do not reflect the traffic's provenance, associated trust, or relationship to other processes or hosts. In this poster, we present the design and prototype of *Pedigree*, a system that tracks the flow of information between files and processes both within and across hosts, and attaches tags which contain this provenance information to network packets generated by processes. Network operators can use tags to realize a rich set of traffic classificaton functions, including fine-grained network-level access control, exfiltration prevention, and identifying malware.

Categories and Subject Descriptors

C.2.2 [**Computer Communication Networks**]: Network Protocols; C.2.0 [**Computer Communication Networks**]: Security and protection

General Terms

Design, Security

Keywords

information flow control, provenance, traffic classification

1. MOTIVATION

Enterprise and transit network operators need to classify and differentiate network traffic to enable provisioning and keep networks secure. Ideally, operators should be able to differentiate traffic according to expressive features, such as the application that generates the traffic, the host or user that generated the traffic, and the associated privileges of that user, whether or not that host might be infected, whether the packets might be carrying potential infections or involved in illegal exfiltration of data, and so forth. Differentiating traffic on such features would allow operators to upgrade or downgrade the service seen by particular traffic flows based on flexible attributes and properties and would, as a result, facilitate much more expressive policies (*e.g.*, filtering traffic based on whether the process that generated the traffic had talked to a known infected host or not). Operators might also want to control traffic based on the properties of the process that generated it (*i.e.*, the application that generated it, or what other hosts or files may have affected the process).

Unfortunately, today, traffic classification remains imprecise. Network operators typically classify traffic using port numbers or IP addresses. This approach is often too coarse or ephemeral. It is also indirect: IP addresses carry no information about the *provenance* of the traffic, such as the process (or group of processes, or host) that generated the traffic. Deploying this type of function is difficult: The large volume of traffic that traverses the network makes inspecting each packet's contents infeasible, and the packets have no markings that bind them to a particular process or group of processes on any particular host. We believe that significant gains in network traffic monitoring require a means to bind that traffic back to a process group (and corresponding level of trust) on a host, and interaction history of the processes that originate packets at the host.

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.
ACM 978-1-60558-175-0/08/08.

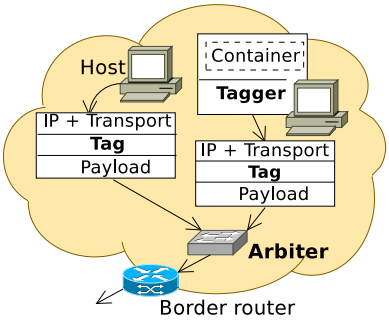

Figure 1: High-level design of *Pedigree*.

2. OUR CONTRIBUTION: PEDIGREE

Pedigree attaches tags to network packets that allows network devices to classify traffic based on the privileges and provenance of that traffic, rather than a coarse identifier like an IP address. *Pedigree* lets network operators express policies based on (1) what *container*—a persistent identifier for a resource (*e.g.*, process group, virtual machine)—generated the traffic; or (2) what other processes, files, or hosts, the process that generated the traffic has interacted with ("taint-set"). *Pedigree* has two components, as shown in Figure 1. The first component is the *tagger*, a trusted module that resides on the host and tags traffic with the identification of the container ("container ID") and taint-set of the process that initiated the traffic. The second module, the *arbiter*, resides on a network element and acts on the traffic according to these tags and the network operator's policy. Network elements can either upgrade traffic (*e.g.*, higher QoS, stronger security guarantees) or downgrade traffic (*e.g.*, drop, blacklist, etc.) based on these tags, which essentially blurs the distinction between the two extreme design points of capabilities (*i.e.*, keeping the network "off by default" and permitting only certain traffic) and filtering (*i.e.*, keeping the network "on by default" and discarding undesirable traffic).

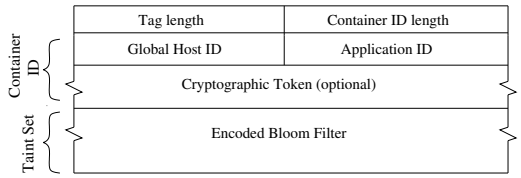

Figure 2: Structure of a tag in *Pedigree*.

2.1 Design and Implementation

Tags. The tag that each packet carries maps to a host container (*i.e.*, process, process group, virtual machine container, etc.) which originated the traffic, and reflects the specific properties of that container, such as whether it has access to certain keys, whether it has been affected by other processes or files (even across hosts), etc. The structure of tag is shown in Figure 2. The containerID contains deterministic fields that provide coarse-grained information about the process originating the packets. The taint-set of a resource—represented using a Bloom filter—comprises the set of identifiers of all the other resources (local or remote) that the resource has interacted with.

Tagger. The tagger is a service residing in the trusted code base (TCB) of a host's operating system. It monitors interactions between resources (processes, files, network sockets, etc.), and whenever data is exchanged between two resources, the reading resource's taint-set is updated (using a set union) with the writing resource's taints. In our prototype, the tagger is a userspace daemon that receives callbacks on system calls. Our preliminary evaluations show that the runtime overhead of intercepting system calls for tracking resource interaction, even at the user level, is between 1 and 4 times the original run time.

Classification. Arbiters in the network take specific actions based on tags in packets: they may use the containerID portion of tags to upgrade service, or use the taint-set portion to deny service (*e.g.*, all packets whose taint-sets have taints corresponding to known malware may be dropped). In addition, because network elements see tags from a wide variety of hosts, they can attempt to cluster various taint-sets in order to identify common taints. Taints may be common, for instance, when many hosts compromised using the same malware are participating in an attack.

2.2 Applications

Pedigree can be used by applications that improve the level of service a certain packet (or flow) receives, such as:

Provisioning. Operators may wish to prioritize certain traffic (*e.g.*, traffic from a messaging application in an enterprise, VoIP packets from an approved application, etc.). *Pedigree* allows operators—especially within enterprises—to easily determine the identity and provenance of the application that generated traffic and take appropriate action.

Access to secure services. Operators may want to authenticate access to certain services or subnets efficiently. Currently, such action is either taken using a static list of IPs, or using application-layer authentication. *Pedigree* allows operators to use fields in the tag, such as the containerID, to ensure legitimacy of the origin application, check for cryptographic signatures, etc.

Similarly, *Pedigree* can also be used to *downgrade* service for flows.

Blacklisting. Today's network-level blacklists (*e.g.*, IP blacklists) need to be continuously updated, are slow in reacting to new malicious hosts, and above all, are based on IPs, which are ephemeral identifiers of hosts. Using *Pedigree*, an arbiter in the network can inspect the taint-sets on the packets from end-hosts to ensure that the sending process has not been affected by known malware by checking the taint-sets of packets it sees for membership of taints of known malware.

Exfiltration Prevention. Data leakage is a critical problem in enterprises. With *Pedigree*, secret data only needs to be "watermarked" by adding one or more unique taints to confidential files' taint-sets. *Pedigree* will ensure that, within the enterprise network, the secret taints are propagated irrespective of whether the files are copied or modified. An arbiter at the edge of the network can prevent data leakage by ensuring that no packet that contains a "watermark" taint is let through.

Host-based security. Similar to ideas such as Process Coloring [1] and Information Flow Control, *Pedigree* can use taint-sets to track resource interactions within a host as well, with the goal of raising alerts when processes with known "bad" taints attempt to modify resources with clean taints, or taints are propagated across processes that violate the information flow criteria.

2.3 Challenges and Limitations

Implementing *Pedigree* entails several challenges.

First, a resource's taint-set may become "polluted" if it reads data from a large number of processes (remote or local), which may be the case with popular web servers. We address this concern by (1) Tracking resources at a finer level (such as at a thread level) such that the taints obtained by a process interacting with other resources are isolated from each other; (2) Using aggressive taint set overflow management schemes such as probabilistically resetting bits of the taint set's Bloom filter.

Second, privacy—a first-order concern these days—may be compromised if hosts append taint-sets to packets sent to unknown hosts (*e.g.*, insecure websites). We counter that the taint-set's Bloom filter is essentially a hash set and the taints in the set cannot be enumerated; the attacker can only ascertain membership of a taint *that he already knows* in an eavesdropped. We are researching ways to secure the taint-set further (*e.g.*, using per-session encryption).

Third, packet overhead may be significant if the taint-sets—often the order of kilobytes—are to be sent on each packet. An immediate optimization (which we have implemented in our prototype) is to send the full taint-set once per flow, but the overhead may still be unacceptable for small flows (*e.g.*, small-volume UDP flows such as DNS lookups). We plan to investigate how sending portions of the taint-set (perhaps on an on-demand basis) affects the overhead and utility of *Pedigree*.

Fourth, tags on the network must be resistant to forgery. *Pedigree* uses re-initializable hash chains (a variant of Lamport's one-way hash chains) to prevent replays.

Finally, lack of widespread adoption may hamper *Pedigree*'s utility. We believe that the benefits of *Pedigree* to ISPs will incentivize them to encourage their users to adopt *Pedigree*.

3. REFERENCES

[1] X. Jiang, A. Walters, F. Buchholz, D. Xu, Y.-M. Wang, and E. H. Spafford. Provenance-Aware Tracing of Worm Break-in and Contaminations: A Process Coloring Approach. In *ICDCS*, June 2006.

[2] A. Ramachandran, K. Bhandankar, M. B. Tariq, and N. Feamster. Packets With Provenance. Technical report, Georgia Tech, May 2008. Georgia Tech CSS Technical Report, available at `http://www.cc.gatech.edu/~avr/pedigree.pdf`.