

# Pedigree: Packets With Provenance

Anirudh Ramachandran, Kaushik Bhandankar, Mukarram Bin Tariq, and Nick Feamster

School of Computer Science, Georgia Tech



## Summary

This project explores ways to use **provenance** information from end hosts to improve network-based traffic classification, and detect and prevent attacks (such as worm outbreaks).

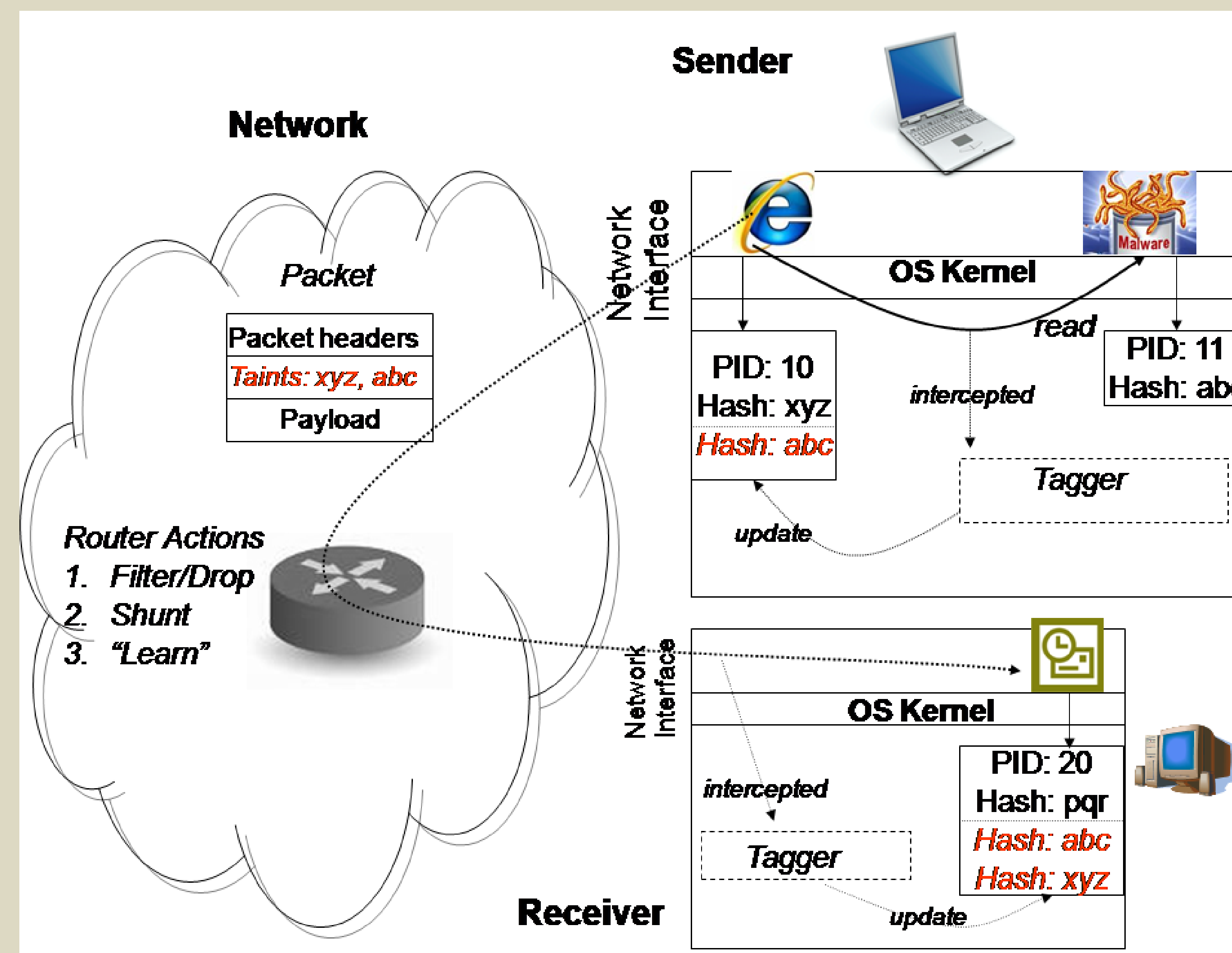
## Motivation

- **Classifying traffic using source/dest IP**: broke
  - DHCP Reallocation
  - IP space hijacking
- **Classifying traffic using port numbers**: broken
  - Easy to change port number
  - Identification is only approximate
- **Deep-packet inspection**: impractical
  - Cannot be performed on high-speed links

## Approach

- Modify end-system software and incorporate a trusted **tagger** which
  1. Maintains “**who-affects-who**” relationships between a host’s internal resources (applications, files, processes, etc.) as well as external ones (network packets, media, etc.)
  2. Attaches **tags** to outgoing network packets with this information

- Augment network devices to read tags in order to
  1. **Filter traffic** from unwanted applications, **detect worm outbreaks**
  2. **Provide service guarantees** for trusted applications
  3. **Exfiltration prevention** for data tagged as “secret”

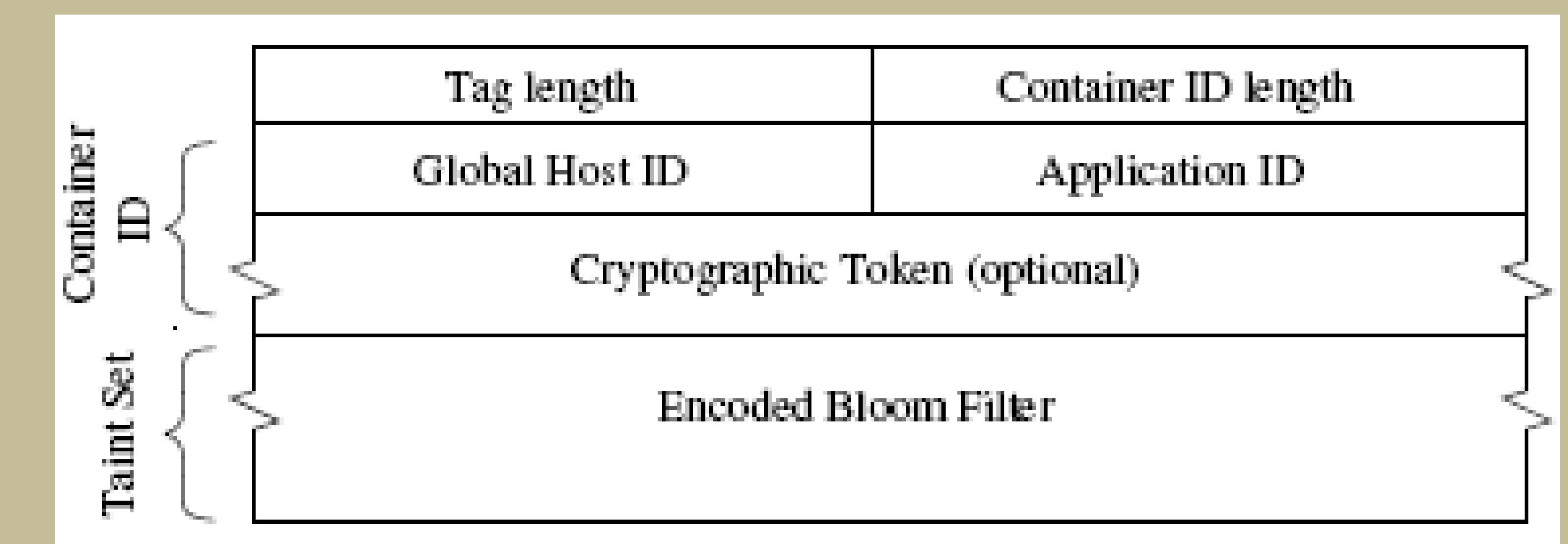


## Implementation

- **System call interception** to track resource interactions
- **User-space implementation of the tagger** (using POSIX message queues)
  - **1.5x-4.5x overhead**
- **In-kernel implementation** in the works

## System Design

- Each resource (file, process) possesses a **tag**
  - **ContainerID**: Deterministic data structure that contains static properties of the application
  - **Taint Set**: Probabilistic structure that stores the **history** of an application’s interactions (represented using a **Bloom Filter**)



- Tags are updated as resources interact (read / write / exec) with each other

## Challenges

- **Packet overhead**: Taint sets need to be large to track a reasonable amount of history
  - Send full taint set only when needed
- **Taint set overflow**: A process that reads a large number of unique resources can pollute its taint set
  - Aggressively purge “old” taints
  - Track taints at thread-level
- **Privacy concerns**: Taints may reveal information
  - Taints cannot be enumerated
- **Lack of widespread deployment**: Pedigree relies on all hosts running a trusted tagger
  - A malicious host could construct fake tags or eavesdrop