Internet Archeology: Estimating Individual Application Trends in Incomplete Historic Traffic Traces

Sebastian Zander, Nigel Williams, Grenville Armitage
{szander,niwilliams,garmitage}@swin.edu.au
http://www.caia.swin.edu.au

Motivation

- Uncover past network application traffic trends
- Available traces usually anonymised and without payload information
  - Payload-based analysis impossible
  - Port-based identification inaccurate for applications such as p2p file-sharing, multiplayer games
- Machine learning (ML) classification based on payload-independent features could be solution
  - Train classifier to detect applications of interest
  - Use classifier on historic traces
Approach

- Obtain representative data for **applications of interest** (positive training examples)

- Can similar applications be separated?
  - 10-fold cross-validation for each trace separately
  - Classes in historic traces based on default ports

- Can recent traffic represent past traffic?
  - Train on recent hand-classified data, test on historic data
  - Train and test between historic data
  - Classes in historic traces based on default ports

Approach cont’d

- Obtain representative data for **all other applications** (negative training examples)
  - Problem: traffic mix in historic trace unknown

- Use ML classifier to identify traffic from historic trace that is not the applications of interest
  - Train classifier with one class for each application plus one class for each port from historic trace
  - Compute *overlap* (false positive/negative rates) between applications of interest and each port
  - If *overlap* > *threshold* ⇒ positive examples
    otherwise ⇒ negative examples
ML Algorithm and Data Sets

- C4.5 decision tree algorithm
- Features: packet length, inter-arrival time, active/idle times, duration, protocol, volume, TCP push
- Traces
  - Payload-classified trace as positive examples (PC)
  - Two public anonymised traces as historic traces (Twente, Leipzig)
- Applications: HTTP/HTTPS, DNS, p2p file-sharing (eDonkey, Kazaa, BitTorrent), game (Half-Life)

Separating Applications

- Each trace separately
- Combine classes of PC and Twente*

* Similar trends for Leipzig [1]
**Predicting Applications**

- **Train on PC, test Twente and Leipzig**
- **Train Twente, test Leipzig and vice versa**

**Estimating Historic Trends**

- **Non-default port numbers Twente**
  (ports >20,000 omitted)
- **ML-based vs. default port estimated traffic volume Twente**

* Similar trends for Leipzig [1]
Conclusions & Future Work

- Similar network applications can be separated
- Application features remain relatively representative between different datasets; but limited variance is problematic
- Approach for obtaining negative examples is somewhat ad-hoc and has limitations; investigate other approaches
- Need historic traces with payload for verification

Tech Report

Poster Arrangement (A1)