Jibber-Jabber? : Encoding the (Un-)Natural Language of Network Devices and Applications

Maxwel Bar-on, Kiley Krosky, Frederico Larrieu
Indrakshi Ray, Indrajit Ray

Colorado State University, USA

Bruhadeshwar Bezawada

Southern Arkansas University, USA

Introduction

- Modern organizational networks have become increasingly complex
- Managing the security of such complex networks is a major challenge
- Accurate identification of the applications and devices connected to a network helps to identify misbehavior and enforce necessary network access controls
- One approach to identify the devices and applications is by analyzing their communication patterns based on the network traffic traces they generate
- Diversity in the networking protocols used by different devices and the complex communication patterns of applications make this a non-trivial task

Problem Statement of Fingerprinting

- **"Fingerprint"**: a meaningful representation of the device/application/threat communication pattern.
- Device/Application/Threat Identification is possible by.
  - Accurate fingerprinting of devices/applications.
  - Matching the fingerprints against recorded fingerprints.

- **Fingerprinting Problem**: to design an algorithm A that generates a fingerprint F with a traffic trace T by considering the packet features, and correlations among the flows in the trace.

- **Matching problem**: accurately compares an unknown fingerprint with the recorded fingerprints to confirm the identification of a device/application/threat.

Limitations of Existing Approaches

Shortage of data

- IoT devices and applications often generate several separate flows to perform a single task
- Existing literature has not explored this hidden correlation of distinct flows generated by the device/application

Related flow semantics

- Many existing approaches focus on analyzing individual traffic flows
- Using flow-based filtering, where each traffic sample only contains packets from a single flow, reduces the amount of noise from unrelated packets
- However, many IoT devices and smart applications do not generate sufficient bi-directional flow data for training accurate machine learning models

Proposed Approach

- We focus on the conversational nature of device communication and model it into a natural language processing (NLP) problem.
  - NLP attempts to design models that can comprehend sentences by processing the natural language words and their correlations.

- We treat a sequence of packets as a collection of correlated words where the correlations are defined by the:
  - Packet content features (e.g., presence of HTTPS, DNS, payload length..)
  - Protocol interactions (e.g., number of DNS queries and responses…)
  - Flows (within a given traffic sample used for fingerprinting)

- A packet is represented as a feature vector that encodes the properties of the packet and its relationships with other packets in a traffic sample

Proposed Approach

- We use machine learning models to analyze sequences of packets and identify the corresponding device or application.

- Our approach solves the two important limitations of previous work:
  - Data shortage due to flow filtering.
  - Correlating distinct flows within a given traffic trace sample.

- To solve data shortage we propose **Window-based sampling**:
  - We consider all packets within a window of traffic rather than filtering based on flows.
  - The window is "sliding", i.e., each packet is viewed as the starting point of the next window, which generates sufficient features for processing.

- To solve relational encoding we propose **Relative Features**:
  - Relative features capture the relationships among packets and flows contained in a window.
  - Our relative features are designed using an embedding scheme based on packet addressing end-points like source IP address, source port etc.,

Key Contributions

- A packet window-based data model to create the analogy to natural language sentences and NLP techniques to solve the problem of network traffic analysis.
  - The “window” may be viewed as a “sentence” of “packets”
- Design of relative features for capturing the inter-dependencies of correlated and uncorrelated flows of a device.
- A multi-class classification technique to address a few network security problems such as
  - IoT device-type fingerprinting,
  - application identification
  - threat identification.
- A comprehensive evaluation of our approach on real-world data sets.

Fingerprinting Details

- Capture window of packets sent to or from an unknown entity over a network interface
  - i.e. IoT device or unknown smart application

- Extract relevant feature vectors from packets in captured window
  - Standard : Features that are extracted from packets.

  - Relative: Extracted from the correlations of the packets using an embedding algorithm of our design.

- Apply deep learning model to feature vectors in window to identify the class of the unknown device
  - We use a Transformer encoder as the deep learning architecture, which is commonly used in natural language processing

Standard Features

Feature	Explanation
Subnet Src	1 if source IP is in subnet else 0
Subnet Dst	1 if destination IP is in subnet else 0
Broadcast Dst	1 if destination IP is broadcast else 0
TLS	1 if packet includes TLS handshake header else 0
TCP	1 if packet includes TCP header else 0
UDP	1 if packet includes UDP header else 0
HTTP	1 if packet uses HTTP port else 0
SSL	1 if packet uses SSL port else 0
Common Src	1 if packet uses common source port else 0
Common Dst	1 if packet uses common dst port else 0
Header Length	length of transport-layer header (bytes)
Payload Length	length of payload (bytes)
Payload Entropy	information entropy of payload

- 13 standard features
- derived independently for each packet in a window
- Mix of categorical and numerical values
- Extracted from network, transport, and payload layers
- Feature design accommodates encrypted packet payloads

Relative Features

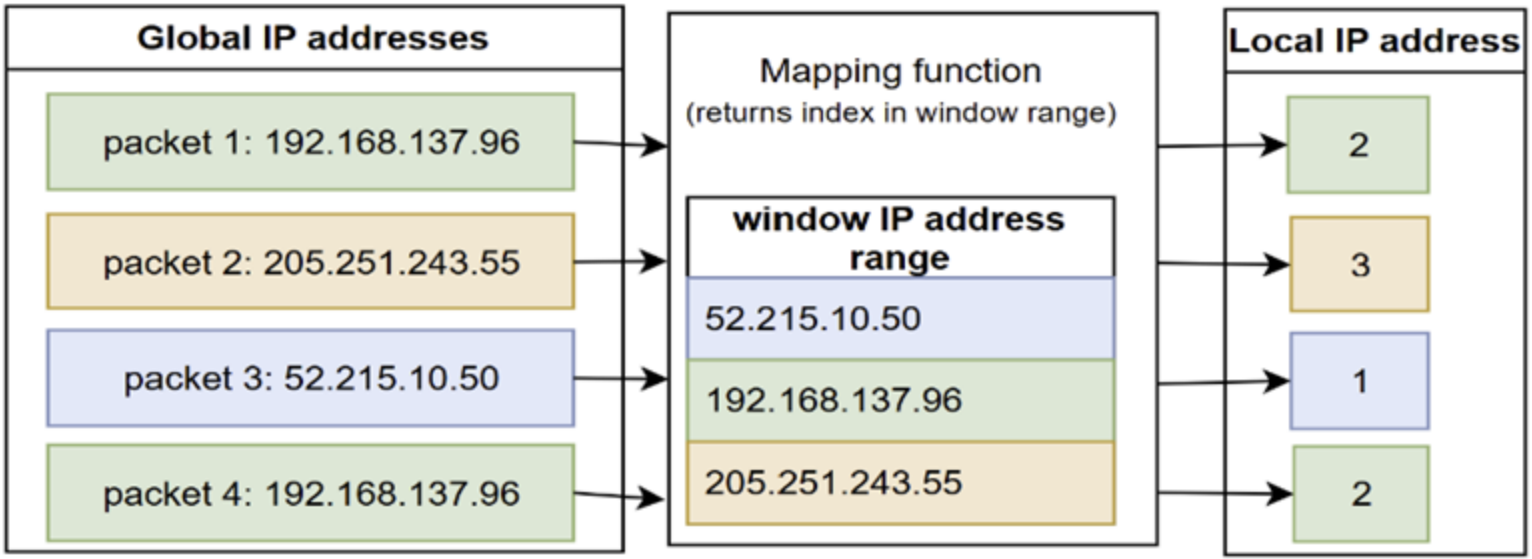
- Encoded representations of the endpoints of packets: <Source IP address, destination IP address, source port, destination port>
  - Useful for capturing relationships among packets and flows within a window
  - Relative features for each packet are represented as 4 integer values

- Generated using relative encoding mapping function
  - Maps each endpoint identifier field to a smaller scale that is local to the packets in a window

- Encoding on a smaller, local scale preserves relationships among packets and flows without causing overfitting by reducing dimensionality and introducing overlap between different classes in the training data

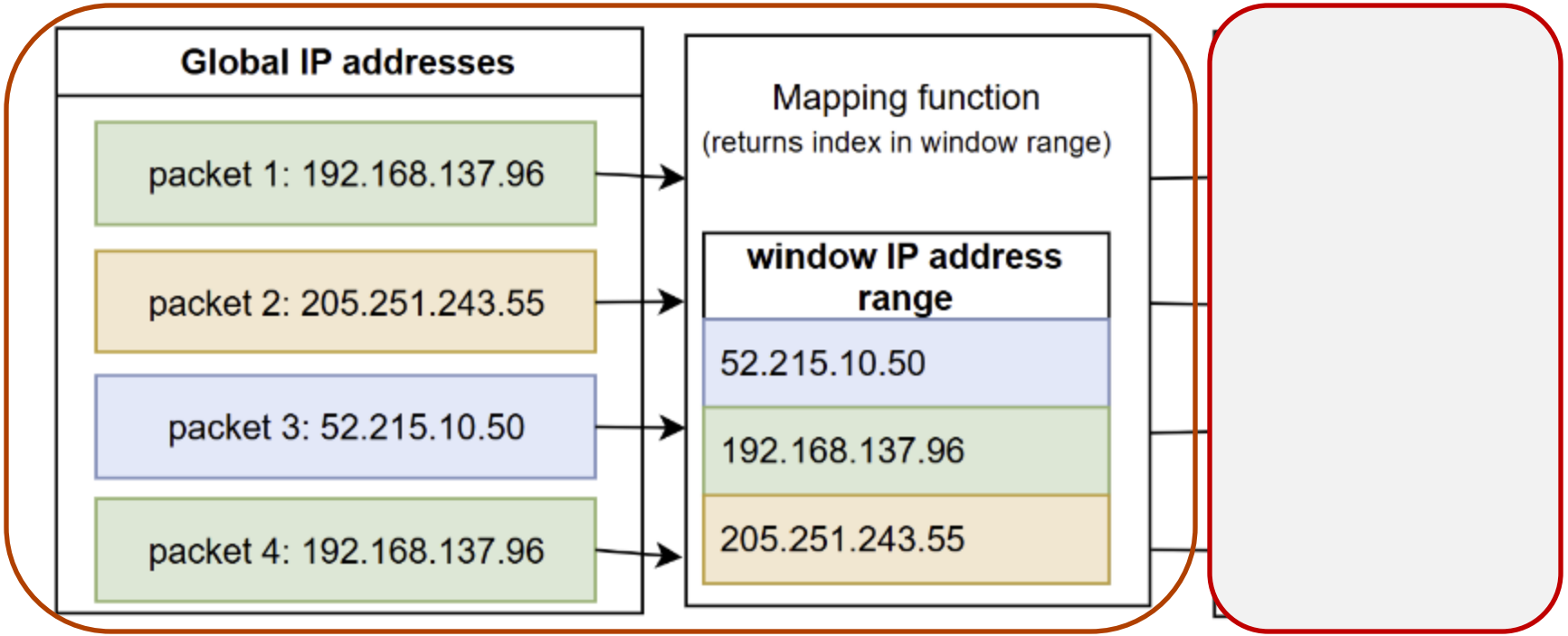
Relative Features: Encoding Process

- Mapping function returns the index of each endpoint identifier field in the sorted list of unique values for the field within the window



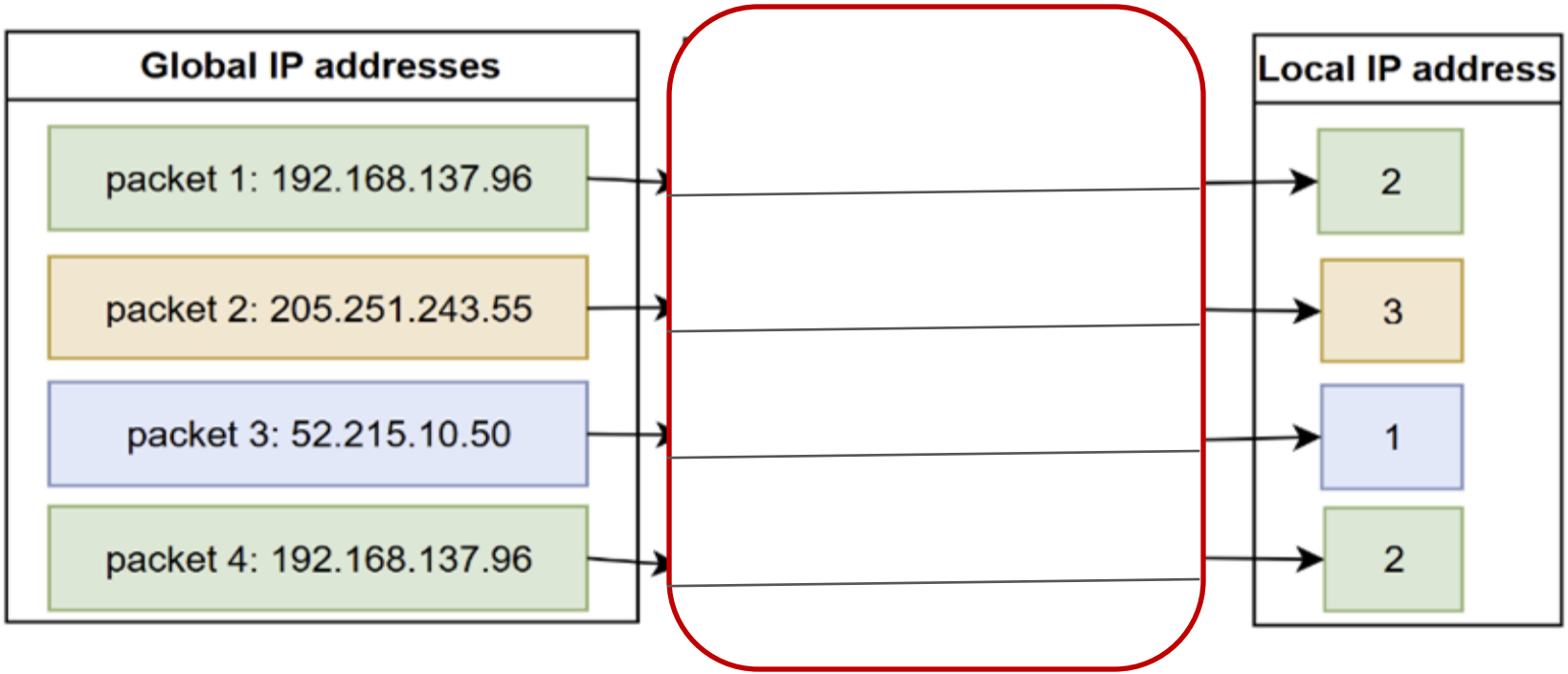
Relative Features: Encoding Process: Step 1

- Mapping function returns the index of each endpoint identifier field in the sorted list of unique values for the field within the window



Relative Features: Encoding Process: Step 2

- Mapping function returns the index of each endpoint identifier field in the sorted list of unique values for the field within the window



- $R\left(P^{(1)} : (192.168.137.96,\ 205.251.243.55,\ 42000,\ 443),\ S\right) = [1, 2, 3, 2]$
- $R\left(P^{(2)} : (205.251.243.55,\ 192.168.137.96,\ 443,\ 42000),\ S\right) = [2, 1, 1, 3]$
- $R\left(P^{(3)} : (192.168.137.96,\ 205.251.243.55,\ 45000,\ 137),\ S\right) = [1, 2, 4, 1]$
- $R\left(P^{(4)} : (192.168.137.96,\ 205.251.243.55,\ 42000,\ 443),\ S\right) = [1, 2, 3, 2]$
- $R\left(P^{(5)} : (192.168.137.96,\ 205.255.4.123,\ 40000,\ 137),\ S\right) = [1, 3, 2, 1]$

	$P^{(1)}$	$P^{(2)}$	$P^{(3)}$	$P^{(4)}$	$P^{(5)}$
$P^{(1)}$	0.4538	0.0083	0.0614	0.4538	0.0226
$P^{(2)}$	0.0171	0.9317	0.0171	0.0171	0.0171
$P^{(3)}$	0.0950	0.0129	0.7021	0.0950	0.0950
$P^{(4)}$	0.4538	0.0083	0.0614	0.4538	0.0226
$P^{(5)}$	0.0397	0.0146	0.1080	0.0397	0.7979

•Relative Features: Illustration using Attention Values

➔ Window of five packets (S)
- ◆ R(packet, S): relative feature mapping function

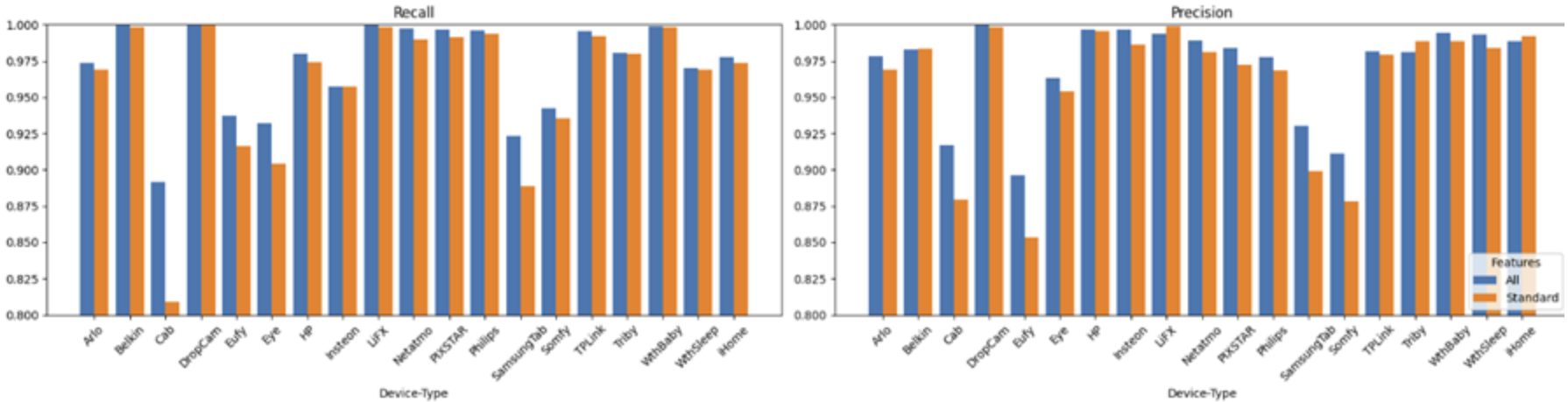
➔ Attention values between packets in S reflect the relationships between the various flows

Experimental Evaluation and Metrics

- Data sets:
  - IoT Fingerprinting: Colorado State University, Canadian Institute of Cybersecurity, and University of New South Wales
  - Mobile Applications and Threats: USTC-TFC (University of Science and Technology China) data set.
- Metrics: We use Precision, Recall, Accuracy and F1-score
- We place higher emphasis on Precision, Recall and F1 because of the necessity to accurately identify the devices/applications/threats.
  - The metric of accuracy is slightly biased towards the performance of the machine learning model and in part depends on the number of data samples under testing and validation

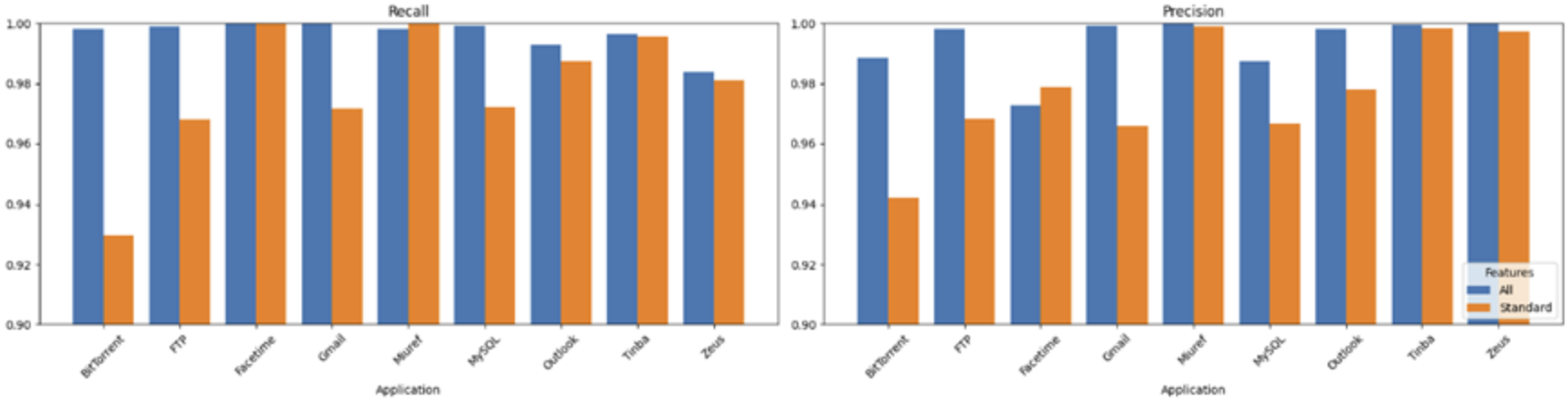
Results: IoT Fingerprinting

- Goal: identify IoT device that generated windows of traffic
- Dataset includes network traffic from 19 different IoT devices
- Results: 97.11% average Recall



Results: Application Identification

- Goal: identify mobile application that generated windows of traffic
- Dataset includes network traffic from 9 different applications
- Results: 99.64% average Recall



Results: Threat Detection

- Goal: distinguish between malicious and benign windows of traffic

		Benign		Malicious	
	Features	all	standard	all	standard
Metric	Recall	0.95	0.94	0.996	0.995
	Precision	0.998	0.997	0.941	0.926
	F1	0.973	0.968	0.968	0.959

Conclusion

- We applied NLP style techniques to network traffic classification and achieved strong performance across three important networking tasks
  - IoT Fingerprinting
  - Application Fingerprinting
  - Threat Identification

- Proposed relative features for capturing relationships between packets and flows in a window of traffic

- Our relative features improved performance across tasks
  - 1.1% average improvement for IoT fingerprinting with up to 10% improvement for complex devices such as “Cab” and “Eufy”
  - 2% average improvement for application identification
  - 0.6% average improvement for threat detection

Future Work

- Transformer models are computationally expensive and time-consuming to test and train.
  - There is need to explore pre-trained models that can amortize this cost.
  - There is need to explore less expensive machine learning architectures.
- The landscape of IoT Devices, Mobile Applications and Digital Threats is evolving and dynamic
  - There are several challenges due to this
- There is need for testing the applicability of existing features to newer data classes.
- Efficiently updating the existing models with new classes is an important future consideration
- Finally, there is need to test for robustness of the machine learning model against adversarial attacks.

Thank you and we welcome any questions!