Design an intelligent ML-based Flow Eviction Algorithm

Group G-11

M Anand Krishna
CS22MTECH14003
IIT Hyderabad

Instructor: Dr. Praveen Tammana, TA: Sankalp Mittal

Apr 27, 2023

Overview

Background

- Importance of flow features in network security
- Few features are packet length, flags, inter-packet timing frequency distributions..
- Using ML to develop efficient flow classification based on the features
- Few challenges on implementing these techniques
  1. Memory contraints : Programmable switches have a limited amount of stateful memory (e.g., 100 MB SRAM), which restricts the amount of data that can be stored and processed in real-time.
  2. Processing Constraints: To ensure line-rate processing, switches have a fixed and short amount of time at each pipeline level for processing packets. This limits the number and type of operations that can be executed within each stage.

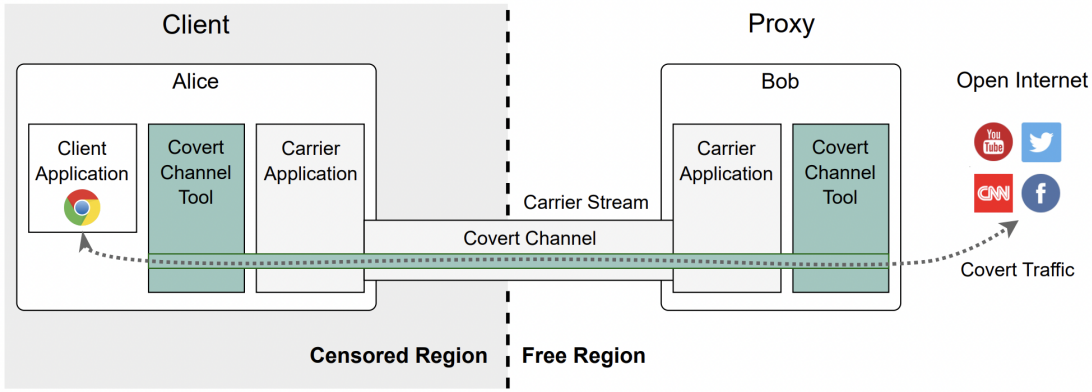
Problem Statement

- The current system effectively classifies malicious flows with high recall only when the number of malicious flows in the training dataset is significantly less than benign flows.
- When the training dataset contains a high percentage (50 or more) of malicious flows, the system's performance could not be as effective, which could lead to reduced recall and accuracy.

Understanding Covert Attack

- A covert channel is a technique used to transmit information between two parties in a way that avoids detection by security mechanisms, such as firewalls, intrusion detection systems, or network monitoring tools

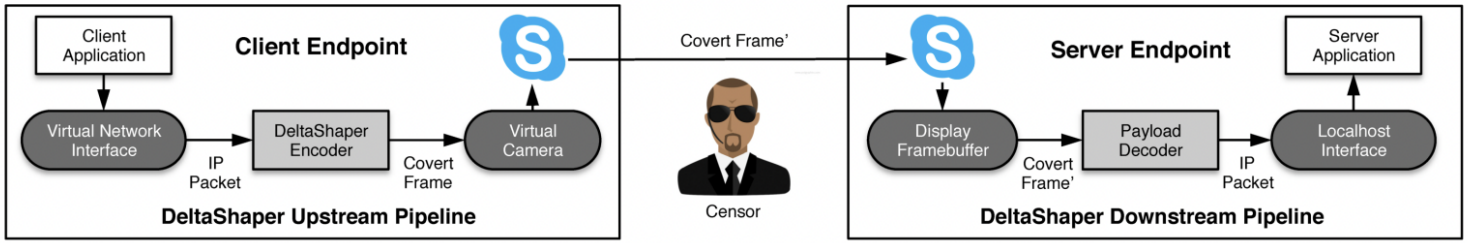
Figure: Basic Idea



DeltaShaper Tool

- DeltaShaper is a censorship-resistant tool that supports bi-directional TCP/IP tunneling over videoconferencing Skype streams.

Figure: Delta Shaper tool



Dataset - DeltaShaperTrafficCaptures

- Benign Traffic -The Author [1] emulated 300 legitimate bi-directional Skype calls by streaming a subset of our legitimate Skype video dataset.
- Malicious Traffic - The Author gathered DeltaShaper traffic samples by establishing a DeltaShaper connection between the Skype endpoints installed in both VMs.

[1] of FlowLens Paper

Work done so far

Understanding Draft Adaflow paper

- AdaFlow with its specially designed cache primitive, which operates within the switch data plane constraints. (both memory and processing)
- AdaFlow minimizes resource overheads and maintains high accuracy in detecting a wide range of network attacks across various networks.

Work done so far - ML phase

- Extracted the Adaflow 11 features from pcap files and labelled them using custom made python script.
- Important feature used to detect Covert attack is average packet size. which is the subset of 11 features because it maximises the recall and accuracy
- Trained a vanilla binary decision tree on the dataset(around 600 pcaps - billion packets).

Performance Metrics of trained DT model

- Accuracy: 69.00 %
- Recall: 97.74 %

Work done so far - Simulation phase

- Reimplemented priority sketch simulation for my understanding

```
        if(IP in pkt):
            #idx = crc16.xmodem(str.encode(str(ip1)+str(ip2)+str(port1)+str(port2)+str(prot)))%size
            idx = crc32.aixm(str.encode(str(ip1)+str(ip2)+str(port1)+str(port2)+str(prot)))%size
            # Case1
            if(start_ts[idx] == 0 and end_ts[idx] == 0 and tuple_reg[idx] == [0,0,0,0,0] and pkt_cnt[idx] == 0 and flow_id[idx] == 0):
                start_ts[idx] = pkt.time
                end_ts[idx] = pkt.time
                tuple_reg[idx] = [srcAddr, dstAddr, srcPort, dstPort, prot]
                pkt_cnt[idx] = 1
                pkt_size[idx] = len(pkt)
                flow_id[idx] = idx
            #Case2
            elif flow_id[idx]!= 0 :
                if flow_id[idx] == idx : #No collision
                    end_ts[idx] = pkt.time
                    tuple_reg[idx] = [srcAddr, dstAddr, srcPort, dstPort, prot]
                    pkt_cnt[idx] += 1
                    pkt_size[idx] += len(pkt)
                    if dt.predict(np.array([[pkt_size[idx]/pkt_cnt[idx]]])) == 1:
                        flow_type[idx] = 1
                        #Eviction Checking
                    if (((pkt.time - end_ts[idx]) > idle_timeout) or ((end_ts[idx] - start_ts[idx]) > active_timeout) or  (TCP in pkt and (pkt['TCP'].flags & FIN or pkt['TCP'].flags & RST))) :
                        with open(output_file_path, "a") as f:
                            line = f"{tuple_reg[idx]},{flow_id[idx]},{pkt_size[idx]/pkt_cnt[idx] if pkt_cnt[idx] != 0 else 0},{flow_type[idx]}\n"
                            case2 +=1
                            f.write(line)
                        start_ts[idx] = 0
                        end_ts[idx] = 0
                        tuple_reg[idx] == [0,0,0,0,0]
                        pkt_cnt[idx] = 0
                        flow_id[idx] = 0
                        flow_type[idx] = 0

                    if flow_id[idx] != idx : #Collision
                        if flow_type[idx] == 0 :
                            with open(output_file_path, "a") as f: #Eviction
                                line = f"{tuple_reg[idx]},{flow_id[idx]},{pkt_size[idx]/pkt_cnt[idx] if pkt_cnt[idx] != 0 else 0},{flow_type[idx]}\n"
                                case3 +=1
                                f.write(line)
                            start_ts[idx] = pkt.time
                            end_ts[idx] = pkt.time
                            tuple_reg[idx] = [srcAddr, dstAddr, srcPort, dstPort, prot]
                            pkt_cnt[idx] = 1
                            pkt_size[idx] = len(pkt)
                            flow_id[idx] = idx
                            flow_type[idx]  = 0 ##??
                        elif flow_type[idx] == 1:
                            case4 +=1
                            continue
    for i in range(size):
        with open(output_file_path, "a") as f: #Eviction
            line = f"{tuple_reg[idx]},{flow_id[idx]},{pkt_size[idx]/pkt_cnt[idx] if pkt_cnt[idx] != 0 else 0},{flow_type[idx]}\n"
            f.write(line)
```

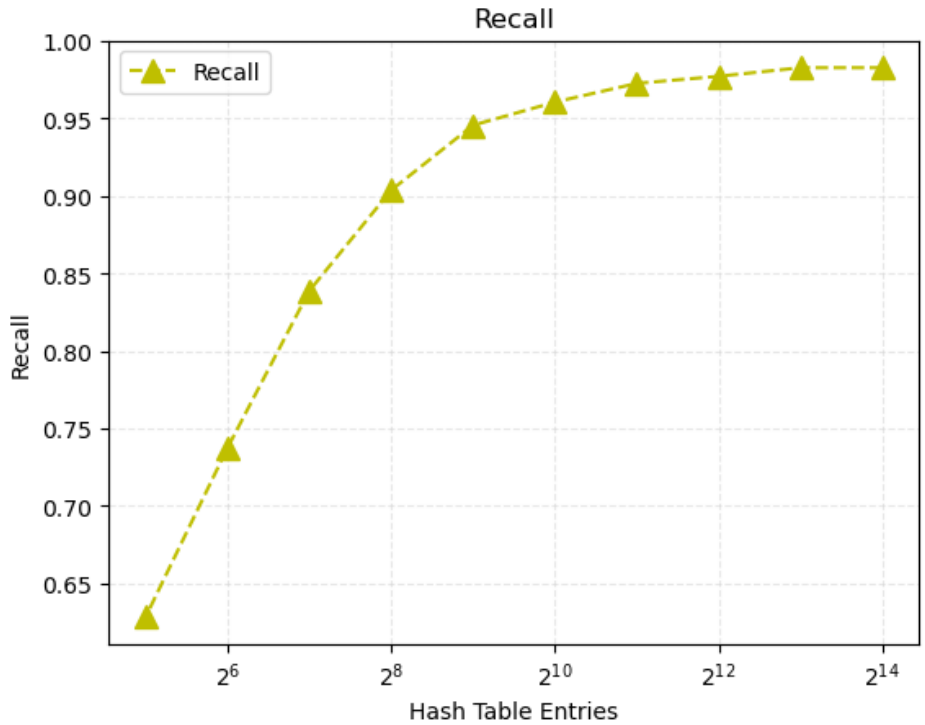
PrioritySketch algorithm

Require: A packet *pkt* with timestamp *ti* of flow *fi*
Ensure: Evicted or updated flow

```
for each packet pkt in pcap flow do
    Extract src_ip, dst_ip, sport, dport, prot from pkt
    Compute index idx using CRC32/16 hash of five tuple
    if cache entry at idx is empty then
        Initialize cache entry with flow information
    else
        if flow ID at idx matches the current flow then
            Update flow information
            if eviction conditions are met (idle and active timeouts, TCP flags) then
                Evict flow and write to output
            end if
        else
            flow ID at idx doesnt match the current flow (Handle collision)
            if existing flow is not malicious then
                Evict and update with new flow
            else
                Ignore the current flow (do not update)
            end if
        end if
    end if
end for
Write remaining flow information to the output file
```

Results

Figure: Recall active timeout = 0.5s and idle timeout = 0.0000001s



Accuracies for all hash table entries around 0.5 ie 50 %

Comparison with existing system NetBeacon

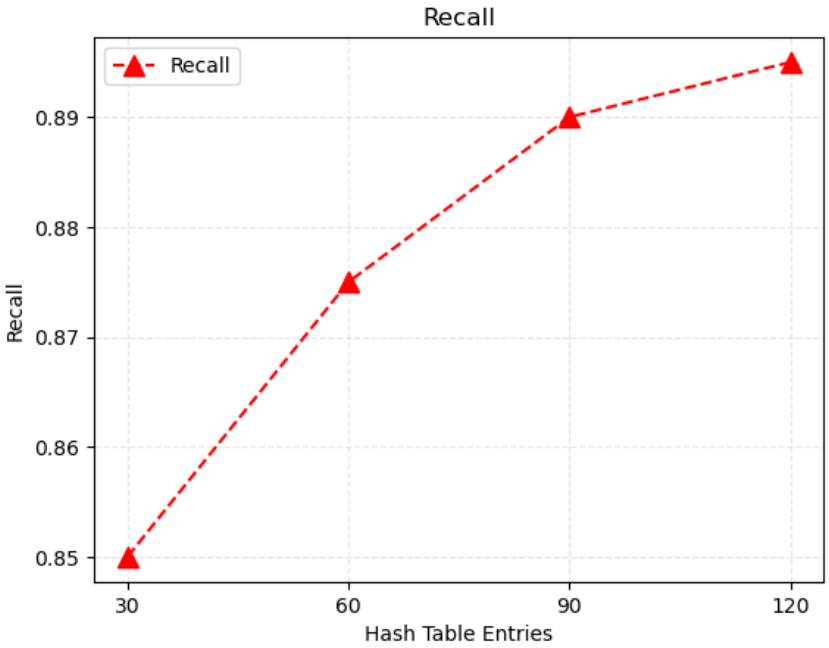


Figure: Recall of NetBeacon

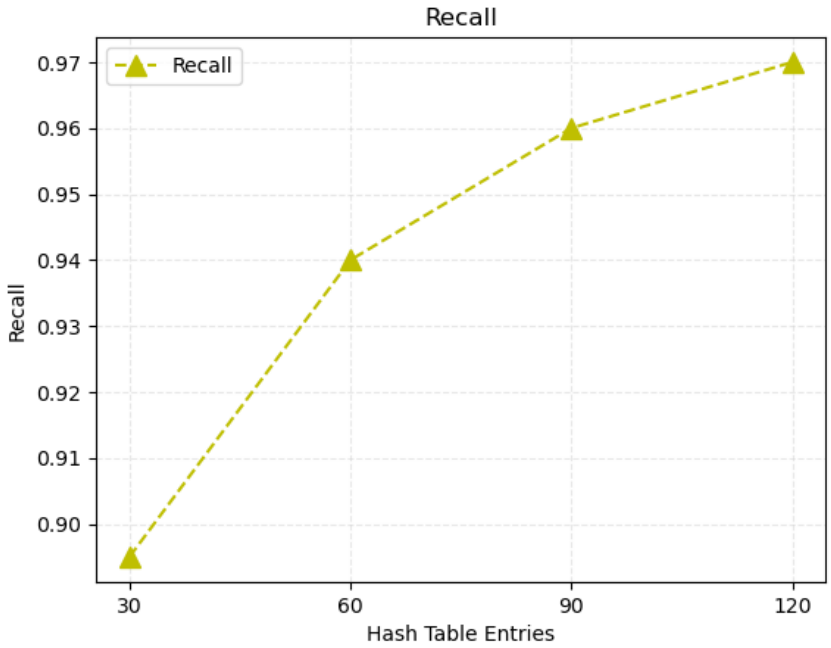


Figure: Recall of our system

Our system is getting better recall result on covert channel detection dataset compared to existing system

References


AdaFlow draft paper


FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications paper

Click here : FlowLens Github


An Efficient Design of Intelligent Network Data Plane paper

THANK YOU


भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad