Generating Labeled Flow Data from MAWILab Traces for Network Intrusion Detection

SNTA2019 Workshop

June 25, 2019

Jinoh Kim, Ph.D.
Computer Science Department
Texas A&M University
Commerce, TX 75428

Introduction

- Increasing attention to the direct identification of malicious activity over network connections
- The boom of the machine learning (ML) industry led to the increasing usage of ML technologies for network intrusion detection
  - To employ ML techniques, datasets are pivotal with the label information to construct learning models
  - However, there exists a shortage of publicly available, relevant datasets to researchers in the network intrusion detection community.
- We introduce a method to construct labeled flow data by combining the ***packet meta-information*** with ***IDS logs*** to promote intrusion detection research
  - Resulted datasets are ***NetFlow-compatible*** including the ***label information***

Intrusion Detection Approaches

- Misuse detection
  - Based on signatures (textual patterns)
  - Accurate to detect known attacks
  - Limited due to:
    - Encryption of packets
    - Legal issue concerning privacy

- Anomaly detection
  - Based on profiling of normal and/or anomalous behaviors
  - Statistical information is used for profiling
    - e.g., duration, number of packets/connection, etc
  - Gained greater attention with ML technologies
  - Data availability is key to succeed!

Challenges for ML-based Anomaly Detection

- Many challenges including the volume of traffic getting heavier than ever (scalability issue)
- Lack of available datasets (containing the associated labels) is another big challenge to employ ML algorithms
- KDDCup 1999 connection dataset has been widely employed but too old!
  - Labels were created by experts with domain knowledge (laborious!)
- We analyze MAWILab traces that provides IDS logs with the packet meta-data to generate labeled flow data.

Data Generation from MAWILab Traces

- Two steps in the generation process:
  - Step 1: Extracting flow information from the packet trace file (pcap)
    - Using SiLK (https://tools.netsa.cert.org/silk/)
  - Step 2: Combining the IDS log data with the flow data constructed in the first step using the four-tuple of flow information
    - Four-tuple: source/destination IP addresses and port numbers

Step 1: generating flow data

- An example trace of “201807011400.pcap” (1426.45 MB for the compressed one)
- Output flow file: “20180701_result.data”

```
> rwptoflow 201807011400.pcap --flow-out=20180701.rw
> rwcut 20180701.rw
  --fields=1,2,3,4,5,6,7,8,9,10,
   11,12,13,14,15,20,21,25,26,27,28,29
  --output-path=20180701_result.data
```

Step 1: generating flow data (cont'd)

- Attributes of flows:
  - Four-tuple: sIP, dIP, sPort, dPort
  - Protocol, pkts, bytes, flags, sTime, duration, eTime, sensor, in, out, nhIP
  - Class, type, icmpTypeCode, initialFlags, sessionFlags, attributes, application

- Reference: https://tools.netsa.cert.org/silk/rwcut.html

Step 2: combining flow data with IDS logs

MAWILab IDS log attributes

Column	Description
sip	Source IP address
dip	Destination IP address
sport	source port
dport	destination port
taxonomy	Category of anomalies (e.g., Port scan, DoS, etc)
heuristic	Code assigned to anomalies using the internal heuristic
distance	$D_n - D_a$, D_n=distance to normal traffic, D_a=distance to anomalous traffic
nbDetectors	Number of detectors reported this anomaly
label	{anomalous, suspicious, notice}

Step 2: combining flow data with IDS logs

MAWILab IDS log attributes

Column	Description
sip	Source IP address
dip	Destination IP address
sport	source port
dport	destination port
taxonomy	Category of anomalies (e.g., Port scan, DoS, etc)
heuristic	Code assigned to anomalies using the internal heuristic
distance	$D_n - D_a$, D_n=distance to normal traffic, D_a=distance to anomalous traffic
nbDetectors	Number of detectors reported this anomaly
label	{anomalous, suspicious, notice}

Combine with flow data based on four tuples!

Step 2: combining flow data with IDS logs – Algorithm

Input: flow_file F, IDS_log R

For each entry F_i in F:
    Search R with 4-tuple in F_i
    If there is a single match with R_j:
        Combine F_i and R_j
        label = anomaly
    If there are multiple matches with $S = \{R_j, R_k, ..\}$:
        Handle multiple match (next slide)
        label = anomaly
    Else:
        label = normal

Step 2: combining flow data with IDS logs – Handling multiple matches

- A log entry may contain null values for certain attributes in 4-tuple
- Define *L* as the number of flow attributes available in 4-tuple (i.e., not null)
- Case 1: *R1*:(sip=A, sport=B, dip=C, dport=D) and *R2*:(sip=A, sport=null, dip=C, dport=null)
  - *L(R1)*=4 > *L(R2)*=2
  - *F1*:(sip=A, sport=B, dip=C, dport=D)
  - *F1* is combined with *R1* by the precedence rule

Step 2: combining flow data with IDS logs – Handling multiple matches

- Case 2: *F2*:(sip=P, sport=Q, dip=R, dport=S), *R3*:(sip=P, dip=R), and *R4*:(dip=R, dport=S)
  - *L(R3) == L(R4)*
- Heuristic:
  1) Give a higher weight to victim than source (i.e., destination > source)
  2) Give a higher weight to host than service (i.e., IP address > port number), and hence (dip > sip > dport > sport) for any identical *L*
- By this rule, *F2* is combined with *R3* instead of *R4*

Step 2: combining flow data with IDS logs – Precedent rule

Priority	# matches	sIP	sPort	dIP	dPort
Highest	4	match	match	match	match
	3	match	null	match	match
	3	match	match	match	null
	3	null	match	match	match
	3	match	match	null	match
	2	match	null	match	null
	2	null	null	match	match
	2	null	match	match	null
	2	match	null	null	match
	2	match	match	null	null
	2	null	match	null	match
	1	null	null	match	null
	1	match	null	null	null
	1	null	null	null	match
Lowest	1	null	match	null	null

Label= anomaly

Label= unsure

- Too many matches for *L=1* log entries => Label the flows as "unsure"
- Example: sport=443 (for secure web browser communication) matches with 23.5% of the flows in total

Example: 12/30/2018 Trace

- Total number of flows: 37M
- Number of anomalous flows: 7.4M (20.1%)
  - Number of bytes for anomalies: 39.4% of the total bytes
- Anomaly classes:
  - Multipoints-class anomalies (57.5%)
  - Network scanning (38.1%)
  - …

Created Data Format

Feature	NetFlow v9 field	Description
sIP	IPV4_SRC_ADDR	Source IP address
dIP	IPV4_DST_ADDR	Dest IP address
sPort	L4_SRC_PORT	Source port
dPort	L4_DST_PORT	Dest port
proto	PROTOCOL	IP protocol
packets	IN_BYTES	Packet count
bytes	IN_PKTS	Byte count
flags	TCP_FLAGS	Bit-wise or of TCP flags over all packets
sTime	UNIX_Seconds	Starting time of flow (in sec)
durat		Duration of flow (in sec)
eTime		End time of flow (in sec)
sen	FLOW_SAMPLER_ID	Name or ID of the sensor
in	SRC_VLAN	Router SNMP input interface
out	DST_VLAN	Router SNMP output interface
nhIP	IPV4_NEXT_HOP	Router next hop ID
senClass		Class of sensor that collected flow (SiLK-specific)
typeFlow		Type of flow for this sensor class (SiLK-specific)
iType	ICMP_TYPE	ICMP type value for ICMP flows
iCode		ICMP code value
initialF		TCP flags on first packet in flow
sessionF		Bit-wise OR of TCP flags over all packets except the first in the flow
attribut		Flow attributes set by the flow generator
appli		Guess as to the content of the flow
class		{normal, anomaly, unsure} for anomaly detection
taxonomy		Category of anomalies (e.g., Port scan, DoS, etc)
label		{normal, anomalous, suspicious, notice} (MAWILab-specific)
heuristic		Code assigned to anomalies (MAWILab-specific)
distance		$D_n - D_a$ (MAWILab-specific)
nbDetectors		Number of detectors reported this anomaly (MAWILab-specific)

Implementation

- Implemented using Python
- `flowlabeling.py` takes a flow data file (resulted in step 1) and an IDS log file, and produces a set of combined flows
- `flowsplitter.py` breaks the outputs into multiple files with designated time windows.
  - For example, it splits a 15-minute flow data into 180 sub-files under the assumption of 5-second time window.
- Available from GitHub repository: https://github.com/dcstamuc/FlowDataGen

Summary

- Introduced a method combining the packet meta-information with the IDS logs to infer labels containing intrusion information for individual network flows.
  - Utilized the SiLK tool to extract the flow data from the TCP dump file
  - Implemented a Python program to combine the flow data with the IDS log.
- The generated flow data contains associated label information for intrusion detection research and is NetFlow compatible.
- The introduced method would assist researchers in network intrusion detection to access recent network flow datasets with associated labels.
- Currently working on the analysis of the constructed data using ML tools For the temporal traffic analysis against the constructed data

THANK YOU!

Questions?

Jinoh.Kim@tamuc.edu