Pentest network traffic classification

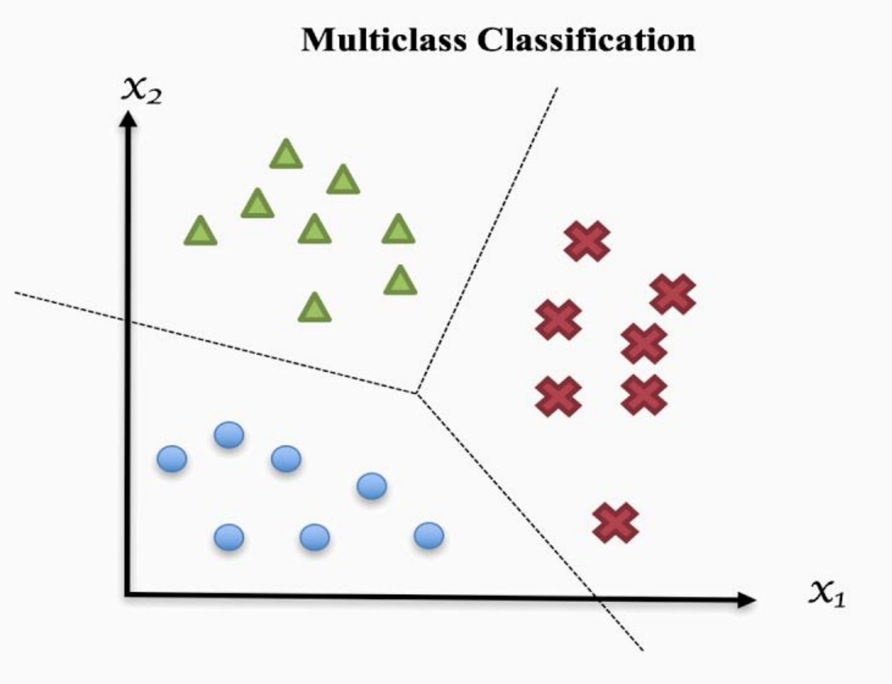
Tiko Huizinga
Supervisors KPMG: Alex Stavroulakis, Soufiane El Aissaoui

Goal

- Pentester needs to log their actions
  - To prove to the client that he performed certain actions
  - To prove to the client that he did not prove certain actions
  - As notes what he did to create a report
- Aid the pentester by automatically log these actions using machine learning
  - Pentester still verifies
- Create a Proof of Concept machine learning tool which:
  - Classifies network traffic using metadata
  - Easy to add new classes of traffic to classifier

Why machine learning?

- Classification problem
- Enough numeric data to train a model
- Manually defining rule based systems requires knowledge about traffic class
    - Machine learning tool only needs example traffic to train



Research question

How reliable is using machine learning in network traffic classification for pentesting auditability?

Method

Gather data	Create PCAP captures per class
Preprocess data	PCAP -> CSV Generalize data & define context Transform to numeric values
Train the classifier	Support Vector Machine (SVM)
Evaluate classifier	Precision, accuracy, F1-score
Repeat process	

Setup

Data generation

- Run tcpdump and filter on target host
- Run one ‘class’ of traffic
- Repeat for each class of data
  - Nmap SYN
  - Nmap ACK
  - Nmap TCP connect
  - Dirb
  - SSH
  - Browsing/other

Pentester

- Run tcpdump on pentesters machine
  - Or machine between tester and testee

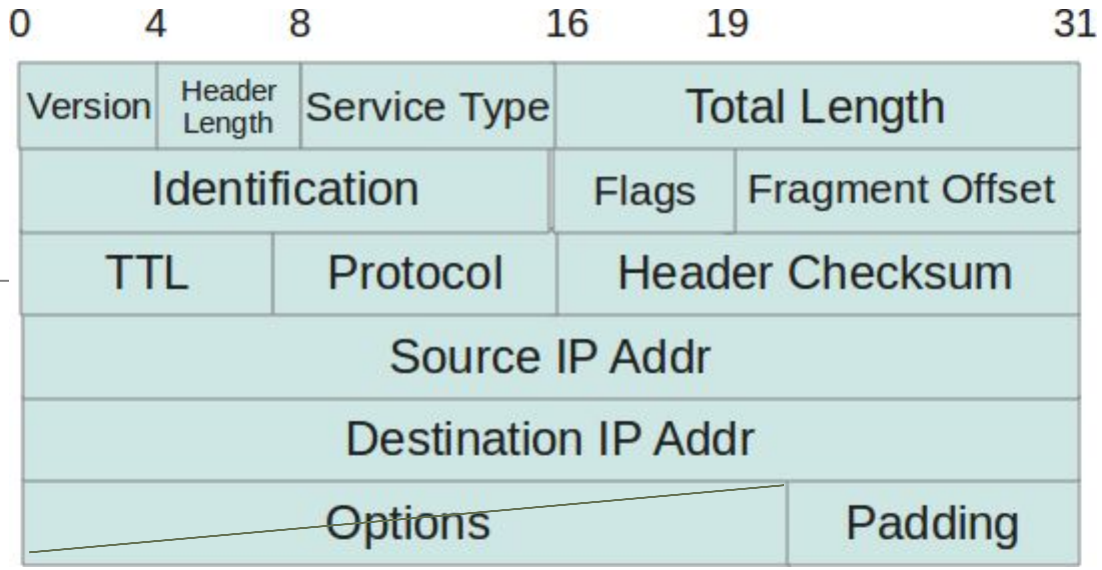
- Execute tests

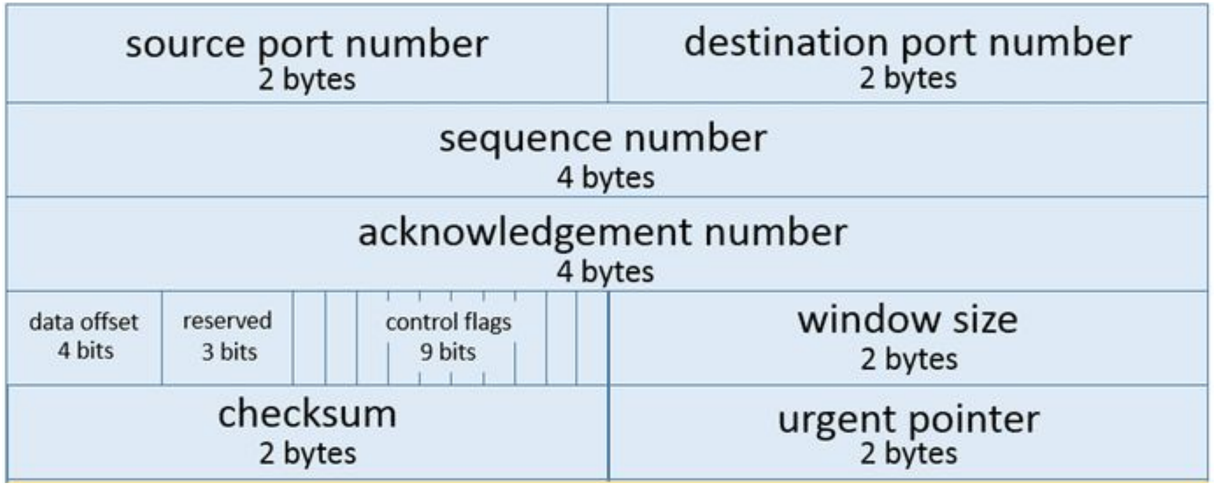
- Submit pcap to tool

- Tool gives list of recognized classes

Parsing pcap

- Parse metadata using Scapy
- Header fields
  - IP header
  - TCP header
- Save field values into CSV





time	ip_version	ihl	tos	ilen	id	iflags	frag	ttl	proto	ichksum	src	dst	ioptions	tsport	tdport	seq
1561115093.8398	4	5	0	28	48620		0	43	1	4184	10.219.188.175	145.100.104.174	[]			
1561115093.83983	4	5	0	44	36911		0	42	6	16128	10.219.188.175	145.100.104.174	[]	33407	443	346274971
1561115093.83985	4	5	0	40	47562		0	54	6	2409	10.219.188.175	145.100.104.174	[]	33407	80	0
1561115093.83986	4	5	0	40	47642		0	53	1	2590	10.219.188.175	145.100.104.174	[]			
1561115093.84444	4	5	96	28	65491		0	58	1	48912	145.100.104.174	10.219.188.175	[]			
1561115094.10416	4	5	0	44	11850		0	51	6	38885	10.219.188.175	145.100.104.174	[]	33663	8888	59502328
1561115094.10421	4	5	0	44	13096		0	54	6	36871	10.219.188.175	145.100.104.174	[]	33663	993	59502328
1561115094.10424	4	5	0	44	16782		0	42	6	36257	10.219.188.175	145.100.104.174	[]	33663	3306	59502328
1561115094.10426	4	5	0	44	20082		0	41	6	33213	10.219.188.175	145.100.104.174	[]	33663	135	59502328
1561115094.10428	4	5	0	44	43134		0	45	6	9137	10.219.188.175	145.100.104.174	[]	33663	1723	59502328
1561115094.10431	4	5	0	44	4113		0	50	6	46878	10.219.188.175	145.100.104.174	[]	33663	587	59502328
1561115094.10433	4	5	0	44	50239		0	44	6	2288	10.219.188.175	145.100.104.174	[]	33663	554	59502328
1561115094.10435	4	5	0	44	45292		0	44	6	7235	10.219.188.175	145.100.104.174	[]	33663	25	59502328
1561115094.10437	4	5	0	44	7361		0	44	6	45166	10.219.188.175	145.100.104.174	[]	33663	113	59502328
1561115094.10439	4	5	0	44	8106		0	41	6	45189	10.219.188.175	145.100.104.174	[]	33663	1025	59502328
1561115095.20563	4	5	0	44	26641		0	49	6	24606	10.219.188.175	145.100.104.174	[]	33664	1025	59495775
1561115095.20823	4	5	0	44	35155		0	53	6	15068	10.219.188.175	145.100.104.174	[]	33664	113	59495775
1561115095.20826	4	5	0	44	31702		0	44	6	20825	10.219.188.175	145.100.104.174	[]	33664	25	59495775
1561115095.20829	4	5	0	44	59602		0	59	6	54620	10.219.188.175	145.100.104.174	[]	33664	554	59495775
1561115095.20831	4	5	0	44	30353		0	54	6	19614	10.219.188.175	145.100.104.174	[]	33664	587	59495775
1561115095.20833	4	5	0	44	17943		0	58	6	31000	10.219.188.175	145.100.104.174	[]	33664	1723	59495775
1561115095.20835	4	5	0	44	61312		0	59	6	52910	10.219.188.175	145.100.104.174	[]	33664	135	59495775
1561115095.20837	4	5	0	44	36830		0	51	6	13905	10.219.188.175	145.100.104.174	[]	33664	3306	59495775
1561115095.20839	4	5	0	44	35746		0	49	6	15501	10.219.188.175	145.100.104.174	[]	33664	993	59495775
1561115095.20841	4	5	0	44	26671		0	49	6	24576	10.219.188.175	145.100.104.174	[]	33664	8888	59495775
1561115095.30592	4	5	0	44	59142		0	40	6	59944	10.219.188.175	145.100.104.174	[]	33663	111	59502328
1561115095.30851	4	5	0	44	34572		0	43	6	18211	10.219.188.175	145.100.104.174	[]	33663	445	59502328
1561115095.30856	4	5	0	44	57159		0	50	6	59367	10.219.188.175	145.100.104.174	[]	33663	256	59502328
1561115095.30858	4	5	0	44	19332		0	52	6	31147	10.219.188.175	145.100.104.174	[]	33663	995	59502328
1561115095.3086	4	5	0	44	10583		0	37	6	43736	10.219.188.175	145.100.104.174	[]	33663	110	59502328
1561115095.30862	4	5	0	44	21623		0	53	6	28600	10.219.188.175	145.100.104.174	[]	33663	80	59502328

Preprocessing

- Generate context
  - Generalize specific fields
  - Number of occurrences in timeframe
- Create only numeric values
  - Not: flags = Syn,Ack
  - But: Flag names → 1 or 0 per flag
- Fill in empty fields
  - Empty port number = 0

Context time = 2 seconds

Time	Dst IP addr
0	8.8.8.8
1	8.8.8.8
2	4.4.4.4
3	8.8.8.8
4	4.4.4.4
5	4.4.4.4
6	3.3.3.3

Same dst IP
1
2
2
1
2
1
0

ip_version	ihl	tos	ilen	id	iflags	frag	ttl	proto	ichksum	same_src	same_dst	tsport	tdport	same_tsport	same_tdport	same_seq	data_class
4	5	0	60	57161	0	0	64	6	39435	340	340	34152	8000	4	279	0	dirb
4	5	0	60	0	0	0	58	6	32597	340	340	8000	34152	277	4	0	dirb
4	5	0	52	57162	0	0	64	6	39442	340	340	34152	8000	4	279	1	dirb
4	5	2	186	57163	0	0	64	6	39305	340	340	34152	8000	4	279	1	dirb
4	5	0	52	48621	0	0	58	6	49519	340	340	8000	34152	277	4	1	dirb
4	5	18	240	30175	0	0	58	6	2224	340	340	22	60898	62	62	0	dirb
4	5	16	52	25615	0	0	64	6	5438	340	340	60898	22	60	60	60	dirb
4	5	2	81	48622	0	0	58	6	49487	340	340	8000	34152	277	4	1	dirb
4	5	0	52	57164	0	0	64	6	39440	340	340	34152	8000	4	279	1	dirb
4	5	2	371	48623	0	0	58	6	49196	340	340	8000	34152	277	4	0	dirb
4	5	0	52	57165	0	0	64	6	39439	340	340	34152	8000	4	279	1	dirb
4	5	0	60	24496	0	0	64	6	6565	340	340	34154	8000	4	279	0	dirb
4	5	0	52	48624	0	0	58	6	49516	340	340	8000	34152	277	4	0	dirb
4	5	0	60	0	0	0	58	6	32597	340	340	8000	34154	277	4	0	dirb
4	5	0	52	24497	0	0	64	6	6572	340	340	34154	8000	4	279	1	dirb
4	5	2	181	24498	0	0	64	6	6440	340	340	34154	8000	4	279	1	dirb
4	5	0	52	28566	0	0	58	6	4039	340	340	8000	34154	277	4	1	dirb
4	5	18	232	30176	0	0	58	6	2231	340	340	22	60898	62	62	0	dirb
4	5	16	52	25616	0	0	64	6	5437	340	340	60898	22	60	60	60	dirb
4	5	2	81	28567	0	0	58	6	4007	340	340	8000	34154	277	4	1	dirb
4	5	0	52	24499	0	0	64	6	6570	340	340	34154	8000	4	279	1	dirb
4	5	2	371	28568	0	0	58	6	3716	340	340	8000	34154	277	4	0	dirb
4	5	0	52	24500	0	0	64	6	6569	340	340	34154	8000	4	279	1	dirb
4	5	0	60	22202	0	0	64	6	8859	340	340	34156	8000	4	279	0	dirb
4	5	0	52	28569	0	0	58	6	4036	340	340	8000	34154	277	4	0	dirb
4	5	0	60	0	0	0	58	6	32597	340	340	8000	34156	277	4	0	dirb
4	5	0	52	22203	0	0	64	6	8866	340	340	34156	8000	4	279	1	dirb
4	5	2	188	22204	0	0	64	6	8727	340	340	34156	8000	4	279	1	dirb
4	5	0	52	53341	0	0	58	6	44799	340	340	8000	34156	277	4	1	dirb
4	5	18	240	30177	0	0	58	6	2222	340	340	22	60898	62	62	0	dirb
4	5	16	52	25617	0	0	64	6	5436	340	340	60898	22	60	60	60	dirb

Training

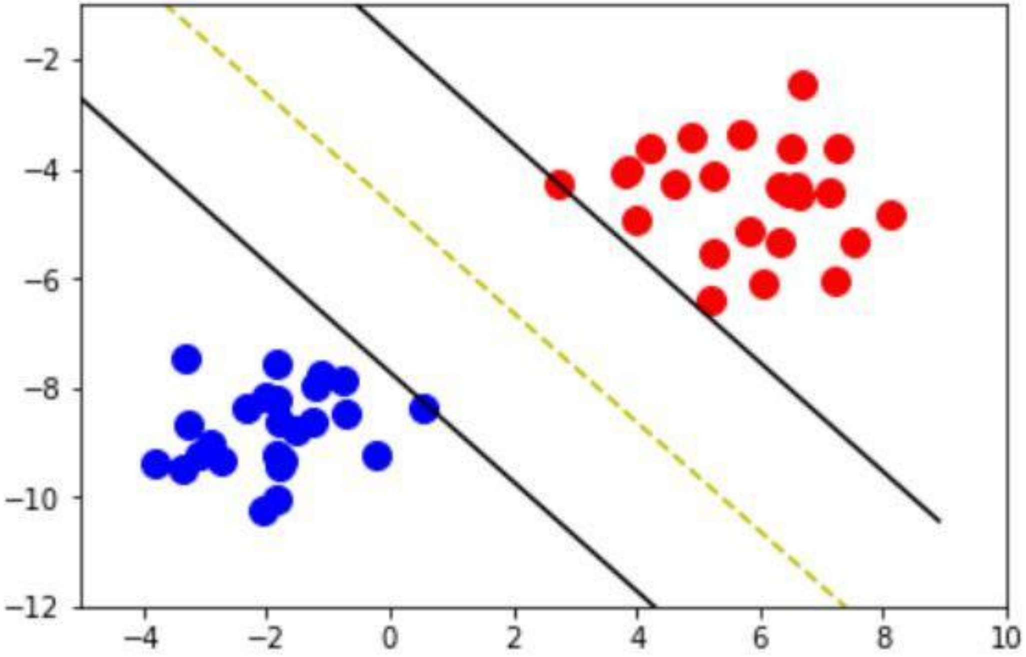
- Classification problem
- Choose suitable algorithm
  - **Support Vector Machine (SVM)**
  - Decision Tree
  - Naïve Bayes
  - Random Forest
  - k-nearest neighbor

Attack	J48 Tree (%)	Naïve Bayes (%)	Random Forest (%)	SVM (%)
loadmodule	0	55.56	33.33	0
rootkit	0	50	10	0
phf	100	75	75	0
buffer_ overflow	70	13.33	83.33	60
ftp_write	0	75	37.5	50
spy	0	100	0	0
multihop	0	42.86	42.86	0
perl	66.67	33.33	66.67	0
warezclient	97.94	47.84	99.31	91.47
nmap	95.24	44.59	97.40	96.54
imap	33.33	91.67	100	83.33
warezmaster	80	90	80	75
normal	99.96	65.22	99.99	99.87

Ali et al. Detailed analysis of network attack detection accuracy (2018)

Support Vector Machine (SVM)

- Divide data in two classes
    - Multiclass SVM uses multiple binary classifiers
- Find hyperplane with largest margin
- Different kernels
    - Linear
    - Polynomial
    - Radial basis function



Metrics and evaluation of classifiers

- Confusion matrix
- Calculate
  - Precision
    - Correct/total predictions for class
  - Recall
    - Correct/total elements in class
  - F1-score
    - Harmonic mean of Precision and Recall

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Predicted class

Actual class

	A	B	C
A	Correct		
B		Correct	
C			Correct

Evaluation of classifiers

SVM: Linear kernel

Class	Precision	Recall	F1-score
dirb	1.00	1.00	1.00
nmap ACK	0.99	1.00	0.99
nmap SYN	1.00	1.00	1.00
nmap TCP connect	1.00	0.99	0.99
SSH	1.00	1.00	1.00
Browsing	1.00	1.00	1.00

SVM: RBF (Radial Basis Function)

Class	Precision	Recall	F1-score
dirb	1.00	0.14	0.25
nmap ACK	0	0	0
nmap SYN	0	0	0
nmap TCP connect	0.18	1.00	0.31
SSH	1.00	0.29	0.45
Browsing	1.00	0.12	0.21

Conclusion

- *How reliable is using machine learning in network traffic classification for pentesting auditability?*
    - Machine learning is very good in recognizing predefined types of traffic

Discussion & future work

Problem	**Solution or future work**
Always returns a predefined class	Research what will happen when classifying ‘unknown’ traffic
Randomly splitting each capture in test and training data might result in overfitting	Capture separate test data during real or simulated pentest
Context is defined with time which may vary in each pentest	Define context based on set number of packets