Comparison of Machine Learning classifiers for malware code detection

Mahsa Rezaei Firuzkuhi
2017

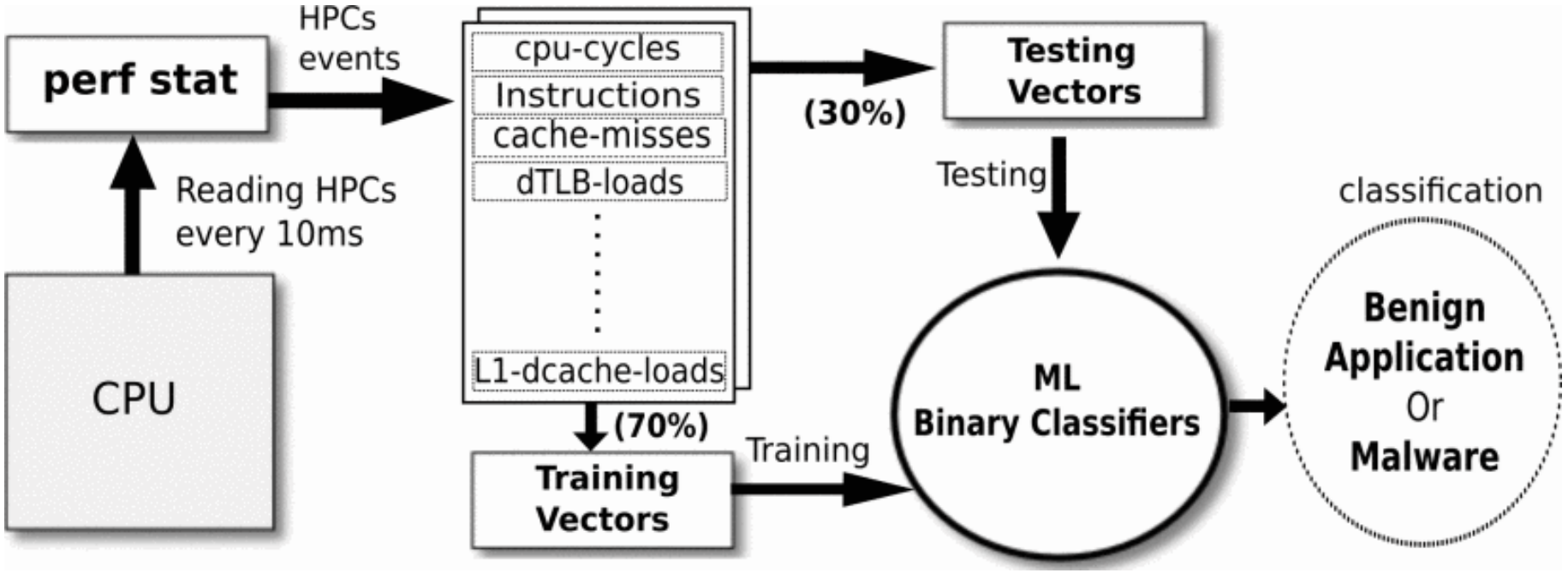
Malware detection using ML

- AV-test.org : In 2017 100+ million new malware. (250,000 per day)
- Antivirus softwares are not robust
- Prone to exploitation
- Detect only static behavior of malware

Introduction

- Antivirus softwares rely on static behavior.
- Antivirus obfuscation techniques :
  - encryption
  - polymorphism
  - metamorphism
  - dead code insertions
  - instruction substitution
- Researches have revealed anomaly in malware execution (J. Demme et al, A. Tang et al)

Main idea of the project



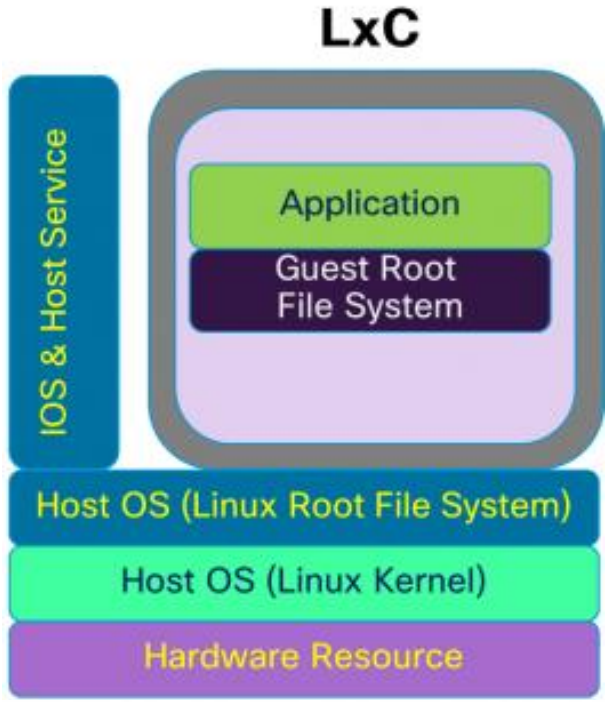
- Hardware-based malware detection trending
- Hardware Performance Counters (HPC) to capture execution pattern
- HPC are integrated in most modern processors

Our approach

- 200 benign and more than 400 malware applications (Virustotal database)
- Intel Haswell Core i5-4590 CPU running Ubuntu 14.04 with Linux 4.4 Kernel
- Linux PERF for collecting HPC values

- Feature table
- WEKA for ML classification

cpu-cycles	instructions	cache-references	cache-misses
branch-instructions	branch-misses	bus-cycles	ref-cycles
cpu-clock	task-clock	page-faults	context-switches
cpu-migrations	minor-faults	major-faults	alignment-faults
dummy	emulation-faults	L1-dcache-loads	L1-dcache-load-misses
L1-dcache-store-misses	L1-dcache-prefetch-misses	L1-icache-load-misses	LLC-loads
LLC-load-misses	LLC-stores	LLC-store-misses	LLC-prefetches
LLC-prefetch-misses	dTLB-loads	dTLB-load-misses	dTLB-stores
dTLB-store-misses	iTLB-loads	iTLB-load-misses	branch-loads
branch-load-misses	node-loads	node-load-misses	node-stores
node-store-misses	node-prefetches	node-prefetch-misses	L1-dcache-stores



Our approach

L1-icache-load-m	branch-loads	LLC-load-misses	L1-dcache-loads	LLC-loads	L1-dcache-stores	L1-dcache-load-r	iTLB-load-misses	bus-cycles	branch-instructio	cache-references	node-loads	branch-misses	node-stores	cache-misses	instructions	class
2394	41551	335	56691	1875	29329	4714	94	600081	1807934	55095	9848	14917	506	11201	38722080	no malware
1856	26384	351	37872	1174	21757	2834	47	833596	2401851	44230	12420	8732	355	13291	55386399	no malware
1262	24310	154	32702	1044	16884	2697	44	241051	816526	48919	3925	14921	307	7470	13620227	malware
1177	20538	168	28222	908	14589	2318	42	99670	917057	42239	2172	21565	345	5259	4469718	no malware
3845	60674	660	87210	2550	50153	6400	88	51788	280280	39941	1799	12847	319	4781	1415795	no malware
4528	77394	616	109385	3880	61733	10064	170	992393	9753549	37028	4276	52074	751	8235	131946761	malware
9932	6101242	2172	17466255	8211	659925	18555	209	1054957	10559860	19482	2479	48876	462	4635	143996897	no malware
9437	6869079	2354	19705646	7202	721633	16241	168	1005721	9302841	26004	3535	54614	590	6811	133519964	no malware
3501	2498142	883	7166123	2672	262653	6001	64	923269	7992376	35702	4847	93280	899	9110	112434289	no malware
15660	837708	2787	2335975	10679	294140	26154	439	322099	2463475	97645	5654	35295	909	12301	30025642	no malware
3120	48119	499	69024	2050	39787	5077	68	952774	7049483	277391	12425	140063	4535	24936	34388687	no malware
3040	45966	539	65632	1979	37941	4780	58	1015542	7897066	272923	23668	139852	3925	35069	44289311	no malware
9802	572389	2017	899011	22914	465168	58717	205	1037346	8223315	266754	17302	119870	2975	25390	52230145	no malware
3780	211511	793	332021	8514	171853	21738	82	1049616	8371113	265965	14199	110827	2548	20656	56090012	no malware
3052	163262	645	256718	6549	134322	16683	70	1042408	8336628	262128	12455	103948	2313	17936	57460922	no malware

- 5 classes of malware: Rootkit, Backdoor, Trojan, Virus, Worm
- 70%-30% data split for training, testing
- Algorithms : Naive Bayes, Logistic Regression, SVM, Multilayer Perceptron, Random Forest, Decision Stump
- Ensemble Voting with performant classifiers
- Measurement : TP rate, and Accuracy

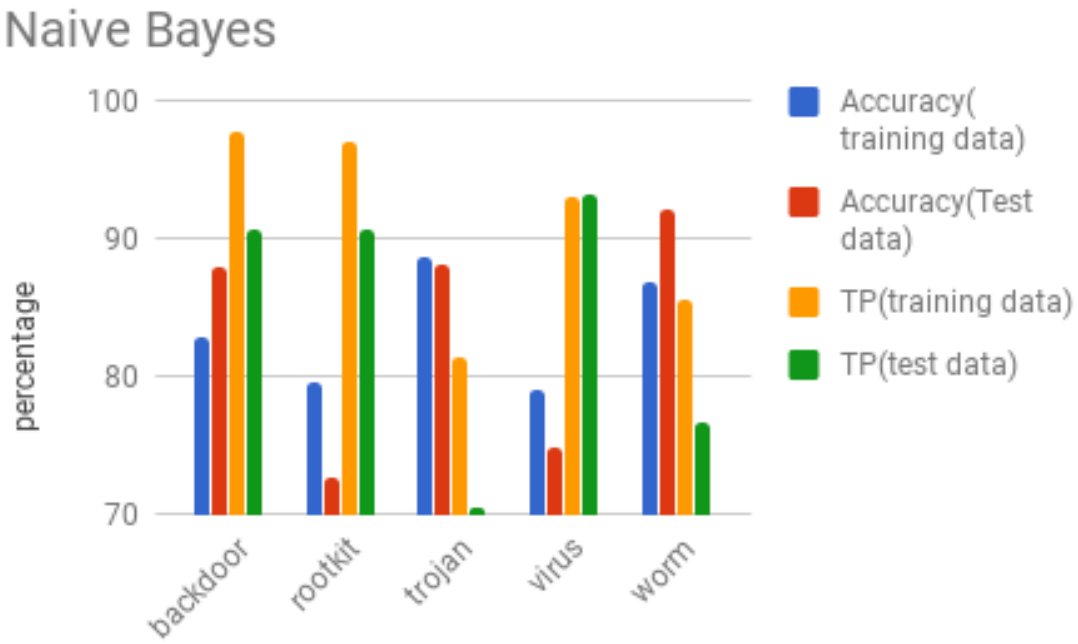
Naive Bayes

- Classifies based on posterior probability
- Considers features as independent

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Results

- rootkit : causing a lot of overhead
- backdoor : Best TP rate of almost 98%, with reasonable overhead



Logistic Regression

Logistic Regression is a modification of Perceptron that has output of probability values between 0 and 1

$$P(y|\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$$

$$\theta(s) = \frac{e^s}{(1+e^s)}$$

Results:

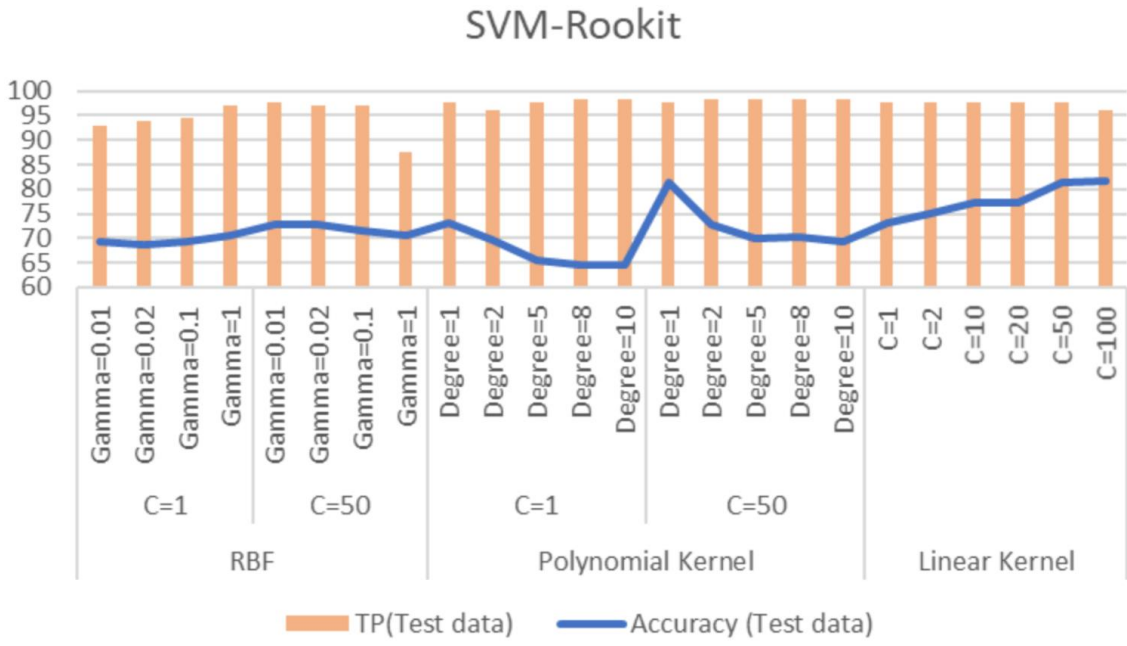
- rootkit : best correlation between TP and Accuracy
- worm : best TP rate with significant FP overhead

Support Vector Machines (SVM)

- SVM Kernels : Linear, Polynomial, RBF

Results:

- Linear : best result among the kernels, hard SVM C=50
- Polynomial : lower degree polynomial C=50
- RBF : Least correlation between TP and Accuracy



Random Forest

- Bagging
- Tree depth 140

Backdoor Results:

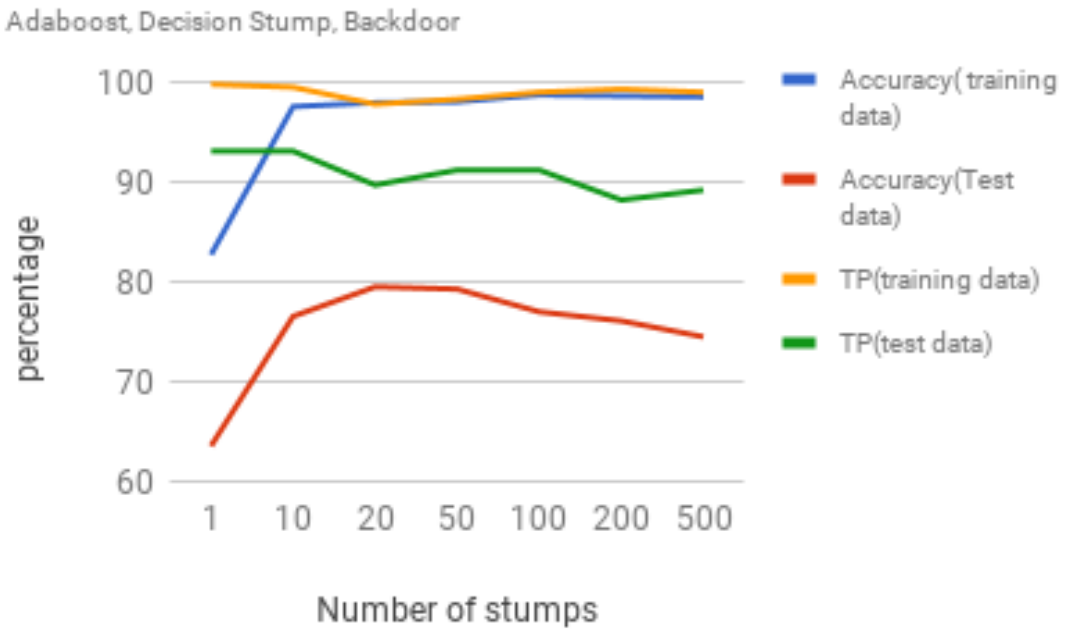
- Overfitting for number of trees greater than 20
- Average overhead of FP

Adaboost(Decision Stumps)

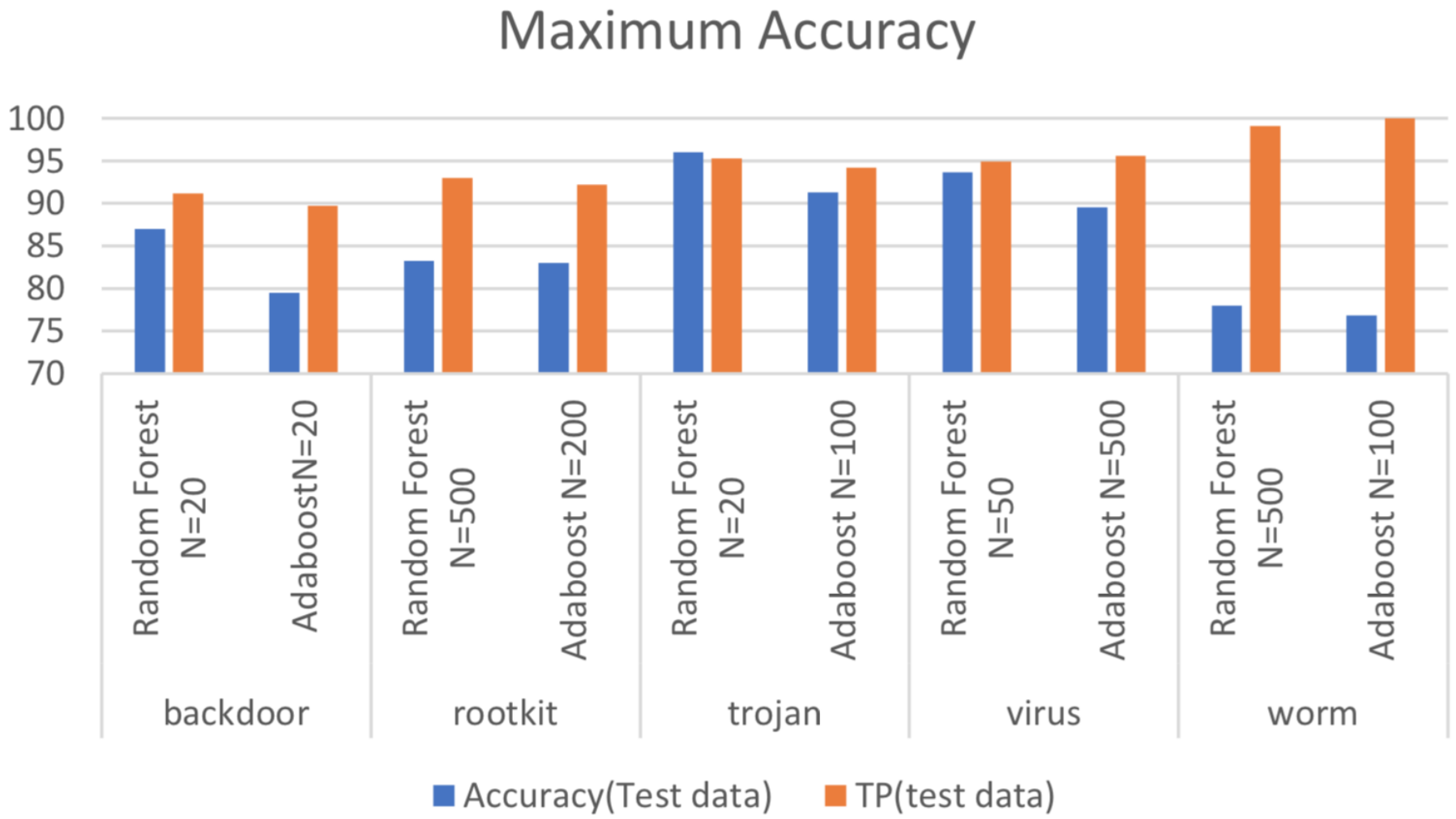
- Decision Stump is a tree of depth 1

Backdoor Results:

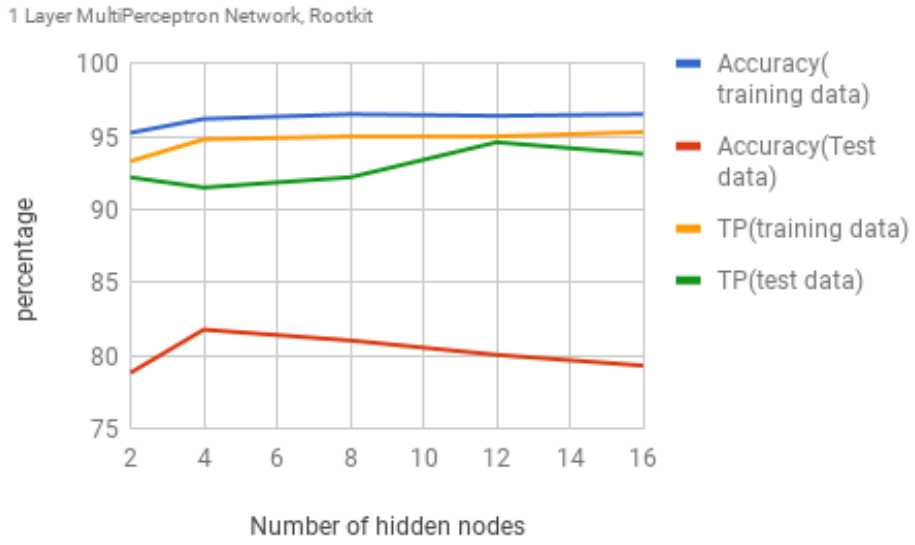
- Overfitting for number of stumps greater than 20
- Overfitting is accompanied by spike of FP overhead

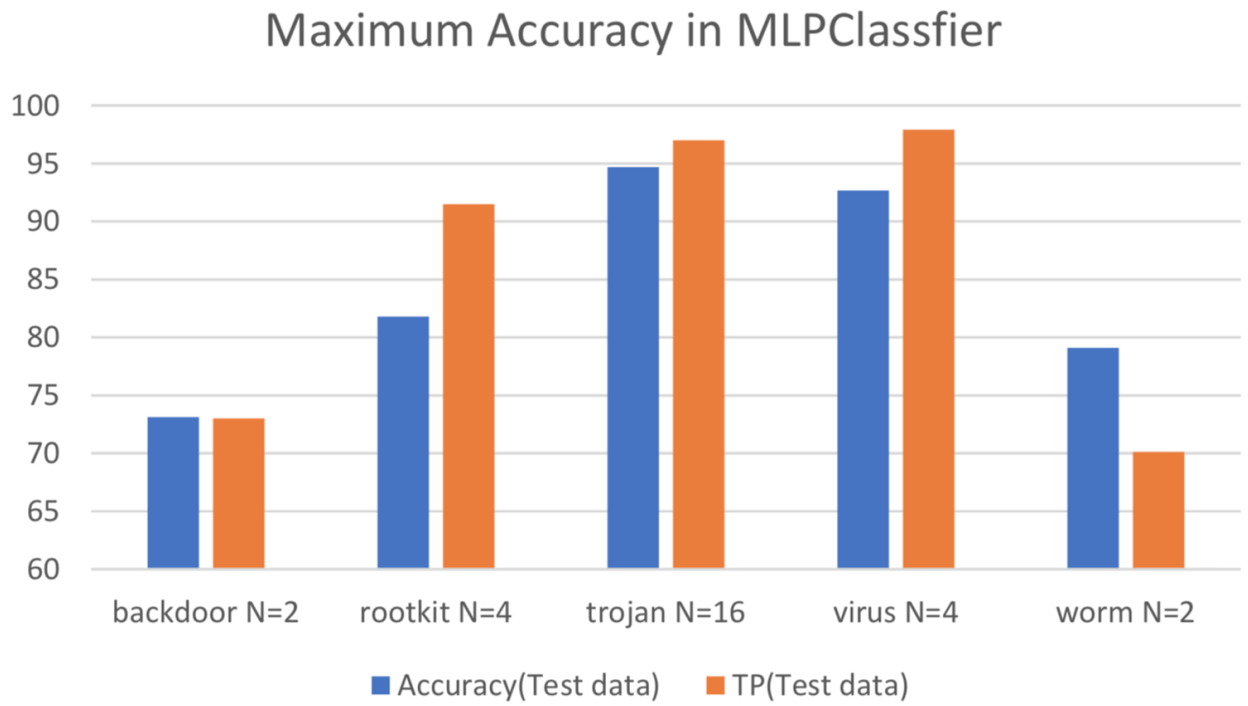


Random Forest vs Adaboost



Multi-layer Perceptron NN





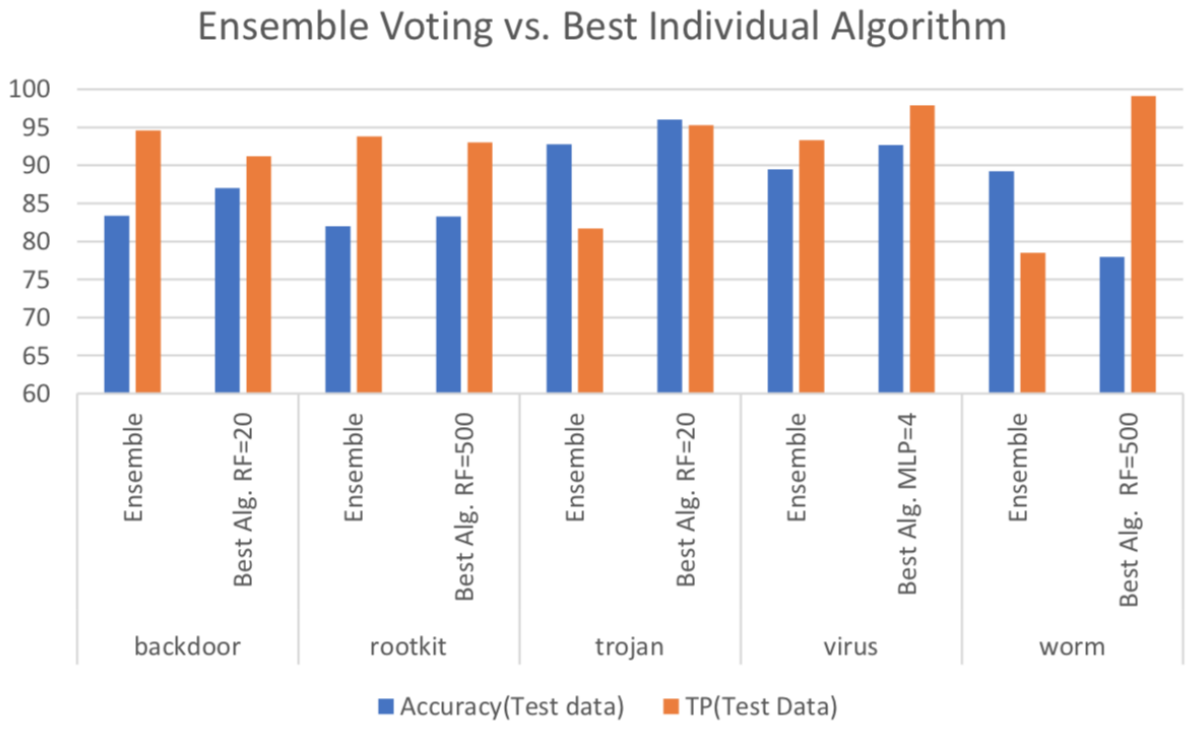
- MLP with 1 hidden layer

Results:

- poor performance for backdoors and worms
- Increasing number of hidden nodes improved performance for trojans

Conclusions

- Overall best performant algorithm is Random Forest
- Ensemble (voting) is comparable to best performing algorithm
- Worms have the lowest detection, which begs more thorough analysis of HPC configuration
- ML based malware detectors together with antivirus can improve detection




TINTA DE ARCHIVE