Intrusion Detection using Outliers in a Cybersecurity Dataset

Nikolaos Perrakis

August 30, 2016

- 4^{th} Industrial Revolution
- Anomaly and Outlier Detection
- Intrusion Detection Systems

DARPA 1998 and KDD Cup 1999 Intrusion Detection Datasets are the first well known attempts to create a solid IDS dataset. However they have many problems[1]:

- Generation procedure could have been more realistic.
- Software used is outdated and misrepresentative of current IT landscape.
- Artefacts of the simulation cause overestimation of efficiency.
- Inconsistency in labels and attacks used.

The main dataset we will use during the MSc Project:

is the **ADFA LD12**[2] Dataset:

- Updated software with the inclusion of a component with a known vulnurenability.
  - Common architectural service of web server solutions (LAMP): Ubuntu 11.04, Apache v2.2.17, PHP v5.3.5 and MySQL v14.14
  - FTP and SSH services were enabled to simulate remote administration
  - Tiki Wiki v8.1 - web based collaborative tool. It has a known vulnerability which simulates 0-day exploitable bug.
- Representative of modern attack structure and methodology.

Attacks used in ADFA-LD 12 dataset.

Payload/Effect	Vector
Password brute force	ftp by hydra
Password brute force	ssh by hydra
Add new superuser	Client side poison executable
Java based meterpreter	Tiki Wiki Vulnerability exploit
Linux meterpreter payload	Client sidepoison executable
C100 Webshell	Php remote file inclusion vulnerability

Data Format

Each data point is a variable length (time) series of kernel system calls!

Subset	Data points
Training	833
Validation	4372
Attack	719

Table: Subsets of ADFA-LD 12 dataset.

- Convert text files in subdirectories to single pickle files.
  Use python dictionary object to maintain all information included in the dataset.
- Perform Exploratory Data Analysis on those files to learn more about the dataset.
  Find which system calls are present on the dataset!

* 325 system calls on kernel but 4 of them are representing by numbers > 325

System calls distribution in ADFA - LD 12

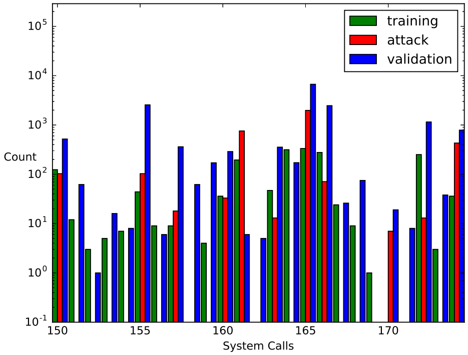
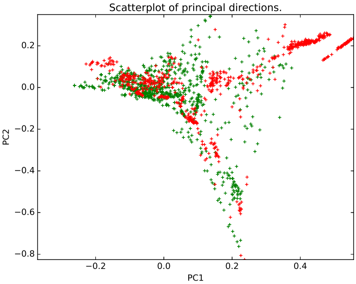
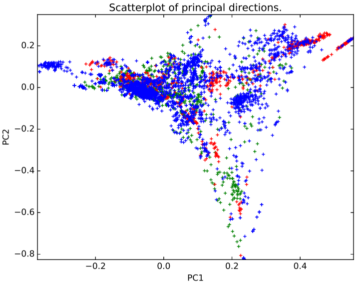


Figure: System calls # 150 - 175 total count in training, attack and validation set.

Frequency Space Principal Components





- Count frequency of each system call on data point to create system calls frequency feature space.
- Perform PCA on the frequency space and keep the first 9 principal components.
- Count frequency of two-sequence system calls on data point to create two-sequence feature space.

Outline

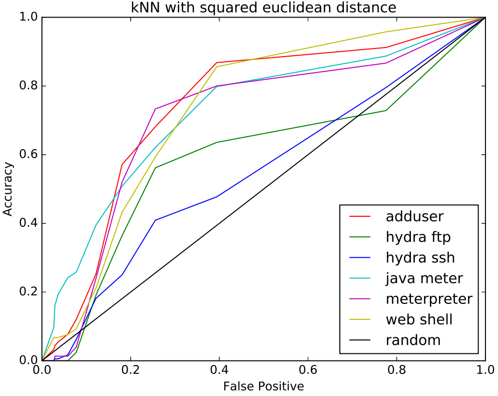
Frequency Feature Space:

- k - Nearest Neighbours[4]
- k - Means Clustering[4]
- Support Vector Machines - Two pattern classification
- One - class Support Vector Machines

Two Sequence Feature Space:

- Support Vector Machines - Two pattern classification
- One - class Support Vector Machines

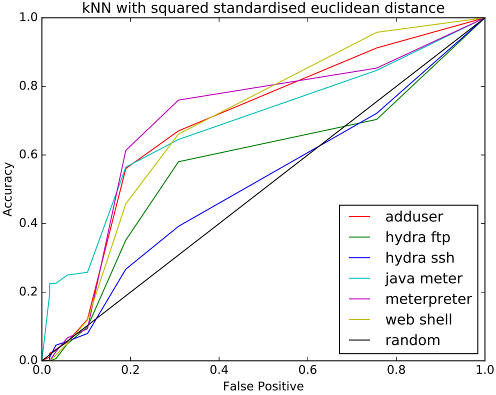
Performance under euclidean distance.



Attack Used	Area under ROC curve
adduser	0.745
hydra ftp	0.593
hydra ssh	0.549
java meterpreter	0.727
meterpreter	0.710
web shell	0.734

Table: Area under the ROC curve.

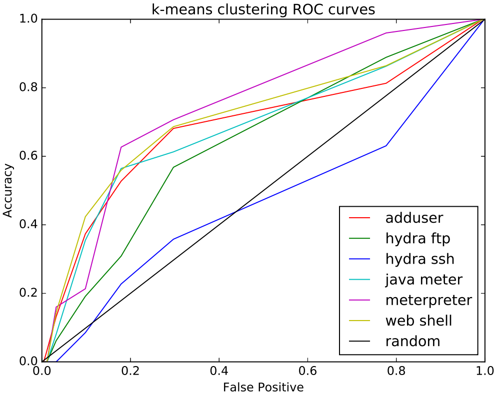
Performance under standardised euclidean distance.



Attack Used	Area under ROC curve
adduser	0.696
hydra ftp	0.574
hydra ssh	0.518
java meterpreter	0.689
meterpreter	0.705
web shell	0.697

Table: Area under the ROC curve.

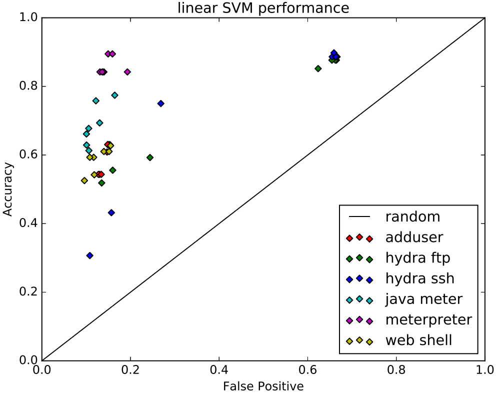
Performance under euclidean distance.



Attack Used	Area under ROC curve
adduser	0.6893
hydra ftp	0.6428
hydra ssh	0.4690
java meterpreter	0.6858
meterpreter	0.7475
web shell	0.7158

Table: Area under the ROC curve.

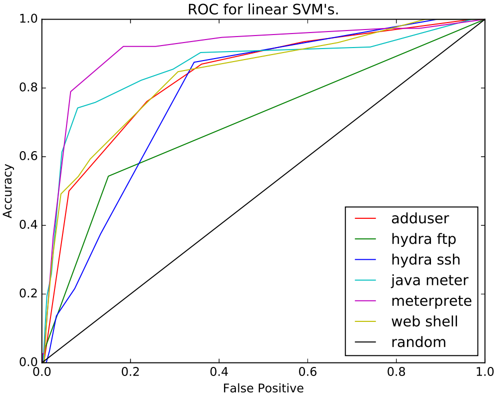
Exploring the SVM parameter space.



Attack Used	Regularisation parameter
adduser	10
hydra ftp	0.05
hydra ssh	0.5
java meterpreter	0.1
meterpreter	10
web shell	1

Table: Optimum regularisation values.

SVM performance with linear kernel.



Attack Used	Area under ROC curve
adduser	0.8303
hydra ftp	0.6978
hydra ssh	0.7801
java meterpreter	0.8628
meterpreter	0.9105
web shell	0.8354

Table: Area under the ROC curve.

SVM on a two pattern classification setting.

Regularisation (C)	Precision	Fall out
0.125	0.62 ± 0.08	0.20 ± 0.03
0.25	0.62 ± 0.08	0.20 ± 0.03
0.5	0.62 ± 0.08	0.19 ± 0.03
1	0.62 ± 0.08	0.19 ± 0.04
2	0.61 ± 0.09	0.19 ± 0.03
4	0.63 ± 0.07	0.20 ± 0.03
8	0.64 ± 0.08	0.21 ± 0.04
16	0.64 ± 0.10	0.23 ± 0.05
32	0.64 ± 0.11	0.24 ± 0.06

Table: 8-fold stratified cross validation results for various regularisation parameter values of SVM classifier with linear kernel. Results are presented with mean and standard deviation for two metrics. True Positive rate (Precision) and False Positive rate (Fall out). Reduced frequency feature space was used for training and validation.

SVM on a two pattern classification setting.

Regularisation (C)	Precision	Fall out
0.125	0.62 ± 0.10	0.17 ± 0.04
0.25	0.64 ± 0.09	0.17 ± 0.04
0.5	0.66 ± 0.09	0.16 ± 0.04
1	0.68 ± 0.06	0.16 ± 0.04
2	0.76 ± 0.07	0.19 ± 0.05
4	0.81 ± 0.08	0.19 ± 0.04
8	0.91 ± 0.04	0.18 ± 0.03
16	0.91 ± 0.05	0.18 ± 0.03
32	0.92 ± 0.04	0.18 ± 0.03
64	0.93 ± 0.05	0.16 ± 0.04
128	0.92 ± 0.06	0.16 ± 0.04

Table: 8-fold stratified cross validation results for various regularisation parameter values of SVM classifier with linear kernel. Results are presented with mean and standard deviation for two metrics. True Positive rate (Precision) and False Positive rate (Fall out). Complete frequency feature space was used for training and validation.

Recursive Feature Elimination with SVM

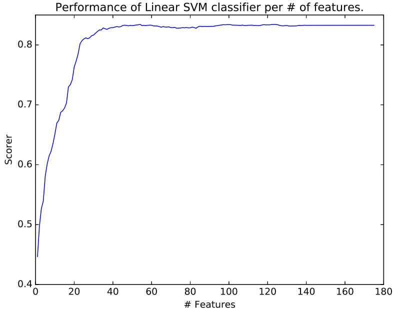
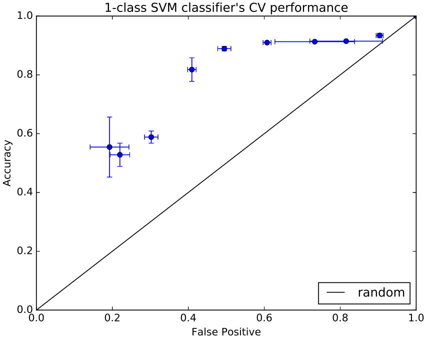


Figure: Scorer (= Precision - Fall out) performance metric compared to number of features in the complete system calls frequency feature space while conducting Recursive Feature Elimination. StratifiedShuffleSplit method was used for cross validation in order to assess scorer. Optimal number of features is 54.

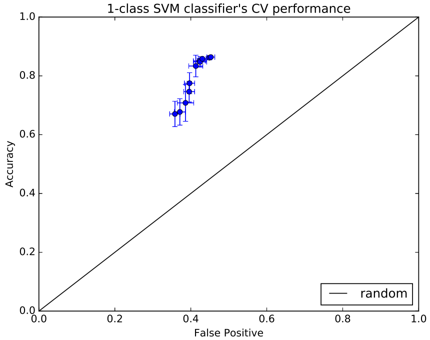
One-class SVM training and bound for training errors.



ν	Precision	Fall out
0.1	0.55 ± 0.10	0.19 ± 0.05
0.2	0.53 ± 0.04	0.22 ± 0.03
0.3	0.59 ± 0.02	0.30 ± 0.02
0.4	0.82 ± 0.04	0.41 ± 0.01
0.5	0.89 ± 0.01	0.49 ± 0.02
0.6	0.910 ± 0.001	0.61 ± 0.01
0.7	$0.912 \pm 1 \cdot 10^{-16}$	0.73 ± 0.10
0.8	$0.91 \pm 6 \cdot 10^{-4}$	0.81 ± 0.10
0.9	0.93 ± 0.007	0.904 ± 0.009
1.0	1.0 ± 0.0	1.0 ± 0.0

Exploring 1-class SVM with a sigmoid kernel to assess it's performance depending on the upper bound for the fraction of training errors. ShuffleSplit method was used for cross validation to create two, equal in size, normal behaviour datasets for training and validation. Optimal performance for upper bound $\nu = 0.4$.

One-class SVM training and bound for training errors.



ν	Precision	Fall out
0.35	0.67 ± 0.04	0.36 ± 0.01
0.36	0.68 ± 0.04	0.37 ± 0.01
0.37	0.71 ± 0.06	0.39 ± 0.02
0.38	0.75 ± 0.04	0.40 ± 0.01
0.39	0.77 ± 0.04	0.40 ± 0.01
0.40	0.83 ± 0.04	0.41 ± 0.02
0.41	0.85 ± 0.01	0.42 ± 0.02
0.42	0.85 ± 0.02	0.42 ± 0.02
0.43	0.858 ± 0.001	0.431 ± 0.010
0.44	0.862 ± 0.002	0.449 ± 0.006
0.45	0.864 ± 0.004	0.45 ± 0.01

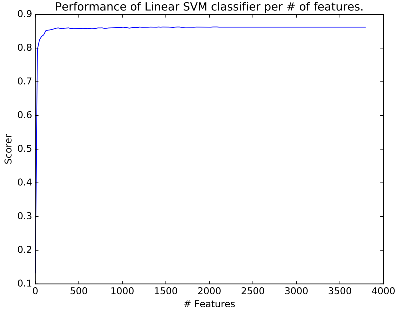
Exploring 1-class SVM with a sigmoid kernel to find it's optimum performance depending on the upper bound for the fraction of training errors. ShuffleSplit method was used for cross validation to create two, equal in size, normal behaviour datasets for training and validation. Optimal performance for upper bound $\nu = 0.41$.

SVM on a two pattern classification setting.

Regularisation (C)	Precision	Fall out
0.125	0.93 ± 0.02	0.068 ± 0.010
0.25	0.93 ± 0.02	0.062 ± 0.009
0.5	0.92 ± 0.02	0.054 ± 0.009
1	0.91 ± 0.02	0.049 ± 0.007
2	0.91 ± 0.02	0.047 ± 0.006
4	0.88 ± 0.03	0.046 ± 0.007
8	0.86 ± 0.03	0.043 ± 0.006
16	0.84 ± 0.04	0.041 ± 0.005
32	0.81 ± 0.03	0.038 ± 0.004

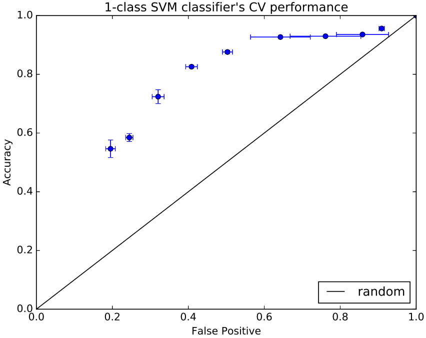
8-fold stratified cross validation results for various regularisation parameter values of SVM classifier with linear kernel. Results are presented with mean and standard deviation for two metrics. True Positive rate (Precision) and False Positive rate (Fall out). Two-sequence feature space was used for training and validation.

Recursive Feature Elimination with SVM



Scorer (= Precision - Fall out) performance metric compared to number of features in the complete system calls two-sequence feature space while conducting Recursive Feature Elimination. StratifiedShuffleSplit method was used for cross validation in order to assess scorer. Elimination step is 24. Optimal number of features is 2088.

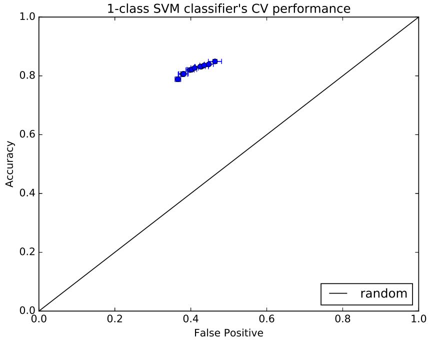
One-class SVM training and bound for training errors.



ν	Precision	Fall out
0.1	0.55 ± 0.03	0.20 ± 0.01
0.2	0.58 ± 0.01	0.245 ± 0.009
0.3	0.72 ± 0.02	0.32 ± 0.02
0.4	0.826 ± 0.004	0.41 ± 0.02
0.5	0.876 ± 0.006	0.50 ± 0.01
0.6	0.9269 ± 0.0008	0.64 ± 0.07
0.7	0.930 ± 0.001	0.76 ± 0.09
0.8	0.936 ± 0.004	0.86 ± 0.07
0.9	0.9560 ± 0.0005	0.909 ± 0.007
1.0	1.0 ± 0.0	1.0 ± 0.0

Exploring 1-class SVM with a sigmoid kernel to assess it's performance depending on the upper bound for the fraction of training errors. Dataset was feature engineered on two-sequence feature space. ShuffleSplit method was used for cross validation to create two, equal in size, normal behaviour datasets for training and validation. Optimal performance for upper bound $\nu = 0.4$.

One-class SVM training and bound for training errors.



ν	Precision	Fall out
0.35	0.788 ± 0.008	0.366 ± 0.007
0.36	0.805 ± 0.008	0.379 ± 0.012
0.37	0.807 ± 0.006	0.380 ± 0.013
0.38	0.820 ± 0.008	0.398 ± 0.011
0.39	0.822 ± 0.003	0.403 ± 0.012
0.40	0.827 ± 0.003	0.410 ± 0.010
0.41	0.832 ± 0.005	0.428 ± 0.018
0.42	0.832 ± 0.004	0.424 ± 0.013
0.43	0.835 ± 0.004	0.435 ± 0.011
0.44	0.839 ± 0.005	0.448 ± 0.012
0.45	0.849 ± 0.006	0.464 ± 0.017

Exploring 1-class SVM with a sigmoid kernel to find it's optimum performance depending on the upper bound for the fraction of training errors. ShuffleSplit method was used for cross validation to create two, equal in size, normal behaviour datasets for training and validation. Optimal performance for upper bound $\nu = 0.36$.

- We successfully replicated results of [4] for kNN and kMC.
- We improved performance with SVM's and by moving to the full feature space.
- We identified key system calls with RFE.
- We demonstrated that moving to two-sequence feature space improves performance.
- We saw that unsupervised learning general purpose approaches do not perform very well.

- Combine domain knowledge with the information provided from recursive feature elimination.
- Improve feature engineering or use more custom kernels for one-class SVM methods for better performance.
- Improve Scalability of algorithms.
  A good candidate is the **AWID 2015**[3] Dataset.
  10 Gb in size, we can check scalability of our methods.

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, Jiankun Hu *A survey of network anomaly detection techniques*, Journal of Network and Computer Applications. Vol. 60, January 2016, p. 19-31

[2] Gideon Creech, Jiankun Huy *Generation of a new IDS Test Dataset: Time to Retire the KDD Collection*, 2013 IEEE Wireless Communications and Networking Conference (WCNC)

[3] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis (2015) *Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset* IEEE Communication Surveys & Tutorials, Vol. 18, No. 1, 2016

[4] M. Xie, J. Hu, X. Yu, and Elizabeth Chang *Evaluating Host-Based Anomaly Detection Systems: Application of the Frequency-Based Algorithms to ADFA-LD*, 11th International Conference on Fuzzy Systems and Knowledge Discovery, 2014

[5] M. Xie, J. Hu and J. Slay *Evaluating Host-based Anomaly Detection Systems: Application of the One-class SVM Algorithm to ADFA-LD*, Proceedings of the 11th IEEE International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2014), Xiamen, 19-21 August 2014, 978-982.