Thirteenth International Conference
Financial and Actuarial Mathematics – FAM-2020

Building a Model for Network Intrusion Detection with Artificial Neural Networks

Daniela Gotseva
dgoceva@tu-sofia.bg
Roumen Trifonov
r_trifonov@tu-sofia.bg
Pavel Stoynov
Technicla University – Sofia,
Faculty of Computer Science and Technology
Sofia, Bulgaria

Abstract: In this article, application of Artificial Neural Networks (ANN) for building Intrusion Detection Systems (IDS) is presented. Data models are necessary and important when building IDS. In the article, the accent is put on feed-forward ANN and anomaly-based IDS where data models are needed to train and evaluate the ANN. Network-based IDS are considered and a statistical approach for building the ANN for intrusion detection is presented.

Key words: Artificial Neural Networks (ANN), IT security, Intrusion Detection Systems (IDS), Data Models, Host-based IDS, R language

Introduction

Feed-forward Artificial Neural Networks (FF ANN) are Artificial Neural networks (ANN) with no feedback loops. In general we may say that a network is feed-forward if it is possible to attach successive numbers to the inputs and to all of the hidden and output units such that each unit only receives connections from inputs or units having smaller numbers (Bishop, 1995).

Intrusion Detection Systems (IDS) are IT security systems for detecting hostile activity. When these systems also prevent the hostile activity, they are called Intrusion Detection and Prevention Systems (IDPS). They are host-based or network based (Gotseva, Trifonov and Stoynov, 2019).

The issue of data selection and preparation when working with neural networks is of high importance. It is crutial for building well adequate models and this motivates the researchers in the area to look for reliable data and for consistent ways for data preparation. The lack of public data sets is a challenges for anomaly-based intrusion detection (Sommer and Paxson, 2010; Haider et al., 2017) and the research community is working on this issue. A good review of the data sets for network intrusion detection is given by (Ring et al., 2019).

In this article, data models for network-based Intrusion Detection Systems (IDS) based on Feed-forward Artificial Neural Networks (FF ANN) are presented.

Data

Some data sets for intrusion detection are considered in Stoynov (2019a, 2019b).

Network-based data appear in packet-based or flow-based format. Flow-based data contain only meta information about network connections, packet-based data also contain payload (Ring et al., 2019).

It is possible to convert packet-based data to flow-based data (but not vice versa) with tools like nfdump 2 or YAF 3 (Haddadi and Zincir-Heywood, 2014).

The data set which we use to build the model is generated by Al-Kasassbeh, Al-Naymat, Al-Hawari (2018). The authors create a realistic dataset with 4998 records for anomaly detection. This dataset was actually based on SNMP variables collected from network devices in a real test network. The variables are grouped in 5 categories: Interface, IP, ICMP, TCP, UDP. The data set is labeled with 8 traffic categories: bruteForce, httpFlood, icmp-echo, normal, slowloris, slowpost, tcp-syn, udp-flood.

The statistics of the output variable – traffic category - is presented in Table 1.

Table 1. Absolute frequency of Execution Path

Traffic category	Frequency
bruteForce	200
httpFlood	573
icmp-echo	632
normal	600
slowloris	780
slowpost	480
tcp-syn	960
udp-flood	773

All input variables are: ifInOctets11, ifOutOctets11, ifoutDiscards11, ifInUcastPkts11, ifInNUcastPkts11, ifInDiscards11, ifOutUcastPkts11, ifOutNUcastPkts11, tcpOutRsts, tcpInSegs, tcpOutSegs, tcpPassiveOpens, tcpRetransSegs, tcpCurrEstab, tcpEstabResets, tcpActiveOpens, udpInDatagrams, udpOutDatagrams, udpInErrors, udpNoPorts, ipInReceives, ipInDelivers, ipOutRequests, ipOutDiscards, ipInDiscards, ipForwDatagrams, ipOutNoRoutes, ipInAddrErrors, icmpInMsgs, icmpInDestUnreachs, icmpOutMsgs, icmpOutDestUnreachs, icmpInEchos, icmpOutEchoReps, class.

To select the input variables which best explane the type of network traffic, construct Table 2, 3 and 4 where for every class of traffic are presented the average numbers of: (tcpInSegs) and output (tcpOutSegs) TCP segments, the input (udpInDatagrams) and output (udpOutDatagrams) UDP datagrams, input IP packets delivered (ipInDelivers), input IP packets received (ipInReceives), input IP packets discarded (ipInDiscards), output IP packets requested (ipOutRequests), output IP packets discarded (ipOutDiscards). The corresponding data are presented in the following tables.

Table 2. Average number of input and output TCP segments by Traffic category.

Traffic category	Average number of input TCP segments (tcpInSegs)	Average number of output TCP segments (tcpOutSegs)
bruteForce	103	101
httpFlood	333	277
icmp-echo	1 159	970
normal	873	694
slowloris	838	677
slowpost	766	629
tcp-syn	580	479
udp-flood	1 261	1 037
All categories	805	660

The data in the Table 2 show that considerable deviation from the normal behavior in terms of TCP segments is observed with bruteForce and httpFlood (negative deviation) and with icmp-echo and udpFlood (positive deviation).

Table 3. Average number of input and output UDP diagrams by Traffic category.

Traffic category	Average number of input UDP datagrams (udpInDatagrams)	Average number of output UDP datagrams (udpOutDatagrams)
bruteForce	180 992	119 937
httpFlood	120 934	88 695
icmp-echo	339 705	267 745
normal	199 647	148 378
slowloris	118 249	75 941
slowpost	206 948	152 275
tcp-syn	245 217	185 956
udp-flood	262 299	194 303
All categories	214 028	158 882

The data in the Table 3 show that considerable deviation from the normal behavior in terms of UDP datagrams is observed with httpFlood and slowloris (negative deviation) and with icmp-echo and udp-flood (positive deviation).

Table 4. Average number of input and output IP packets (delivered, received, requested, discarded) by Traffic category.

Traffic category	Average number of input IP packets delivered - ipInDelivers	Average number of input IP packets received - ipInReceives	Average number of input IP packets discarded - ipInDiscards	Average number of output IP packets requested - ipOutRequests	Average number of output IP packets discarded - ipOutDiscards
bruteForce	186 708	273 848 108	19	120 052	852
httpFlood	123 427	66 331 387	24	88 996	58
icmp-echo	346 072	166 195 750	22	268 651	505
normal	203 607	46 836 956	30	149 155	383
slowloris	122 119	39 674 948	30	76 671	109
slowpost	212 981	160 545 656	39	152 967	79
tcp-syn	250 301	149 355 523	44	186 479	128
udp-flood	268 131	148 117 047	31	195 340	279
All	218 884	118 407 273	32	159 570	243

Table 5. Ranges defined for input TCP segments (variable tcpInSegs).

Average number of input TCP segments (tcpInSegs)	Code
0-450	1
450-900	2
900-1350	3
>1350	4

The data in the Table 4 show that considerable deviation from the normal behavior in terms of input IP packets delivered is observed with httpFlood and slowloris (negative deviation) and with icmp-echo and udp-flood (positive deviation).

In terms of input IP packets received, deviation from the normal behavior is shown by bruteForce, icmp-echo, slowpost, tcp-syn and udp-flood (in positive direction).

From these variables we select as input variables for the ANN the variables: tcpInSegs, udpInDatagrams, ipInReceives, ipOutRequests. To use variables we transform them in ranges.

Table 6. Average number of input TCP segments and their ranges by Traffic category.

Traffic category	Average number of input TCP segments (tcpInSegs)	Range
bruteForce	103	1
httpFlood	333	1
icmp-echo	1 159	3
normal	873	2
slowloris	838	2
slowpost	766	2
tcp-syn	580	2
udp-flood	1 261	3
All categories	805	2

Table 5 shows the transform of the input variable tcpInSegs into ranges.

In Table 6, the defined ranges from Table 5 are mapped to the average number of the input TCP segments for every traffic category. The normal range is 2. Negative deviation (range 1) have bruteForce and httpFlood categories. Positive deviation (range 3) have icmp-echo and udp-flood traffic categories.

Table 7 shows the transform of the input variable udpInDatagrams into ranges.

Table 7. Ranges defined for input UDP datagrams (variable udpInDatagrams).

Average number of input UDP datagrams (udpInDatagrams)	Code
0-125 000	1
125 000-250 000	2
250 000-375 000	3
>375 000	4

In Table 8, the defined ranges from Table 7 are mapped to the average number of input UDP datagrams for every traffic category. The normal range is 2. Negative deviation (range 1) have httpFlood and slowloris categories. Positive deviation (range 3) have icmp-echo and udp-flood traffic categories.

Table 8. Average number of input UDP datagrams and their ranges by Traffic category.

Traffic category	Average number of input UDP datagrams (udpInDatagrams)	Range
bruteForce	180 992	2
httpFlood	120 934	1
icmp-echo	339 705	3
normal	199 647	2
slowloris	118 249	1
slowpost	206 948	2
tcp-syn	245 217	2
udp-flood	262 299	3
All categories	214 028	2

Table 9 shows the transform of the input variable ipInReceives into ranges.

Table 9. Ranges defined for input IP packets received (variable ipInReceives).

Average number of input IP packets received -ipInReceives	Code
0-100 000 000	1
100 000 000-200 000 000	2
200 000 000-300 000 000	3
>300 000 000	4

In Table 10, the defined ranges from Table 9 are mapped to the average number of input IP packets received for every traffic category. The normal range is 1. Moderate positive deviations (range 2) have icmp-echo, slowpost, tcp-syn and udp-flood traffic categories. Large positive deviation (range 3) has bruteForce.

Table 11 shows the transform of the output IP packets requested into ranges.

In Table 12, the defined ranges from Table 11 are mapped to the average number of output IP packets requested for every traffic category. The normal range is 2. Negative deviations (range 1) have httpFlood and slowloris traffic categories. Positive deviations (range 3) has icmp-echo traffic category.

Table 10. Average number of input IP packets received and their ranges by Traffic category.

Traffic category	Average number of input IP packets received - ipInReceives	Range
bruteForce	273 848 108	3
httpFlood	66 331 387	1
icmp-echo	166 195 750	2
normal	46 836 956	1
slowloris	39 674 948	1
slowpost	160 545 656	2
tcp-syn	149 355 523	2
udp-flood	148 117 047	2
All categories	118 407 273	2

Table 11. Ranges defined for output IP packets requested (variable ipOutRequests).

Average number of output IP packets requested -ipOutRequests	Code
0-100 000	1
100 000-200 000	2
200 000-300 000	3
>300 000	4

Table 12. Output IP packets requested and their ranges by Traffic category.

Traffic category	Average number of output IP packets requested -ipOutRequests	Range
bruteForce	120 052	2
httpFlood	88 996	1
icmp-echo	268 651	3
normal	149 155	2
slowloris	76 671	1

slowpost	152 967	2
tcp-syn	186 479	2
udp-flood	195 340	2
All categories	159 570	2

As output variables we select traffic category. The encoding of this variable is given in Table 13.

Table 13. Encoding of the output variable Traffic category.

Traffic category	Code
bruteForce	1
httpFlood	2
icmp-echo	3
normal	4
slowloris	5
slowpost	6
tcp-syn	7
udp-flood	8

The final data model which we construct is a Feed-forward ANN. It has 16 binary input variables presented in Table 14 and 8 output variables y_1 to y_8 which are correspondingly the different traffic categories presented in Table 13.

Table 14. Input variables.

Variable	Observed variable	Range
x_1, x_2, x_3, x_4	input TCP segments	1,2,3,4,
x_5, x_6, x_7, x_8	input UDP datagrams	1,2,3,4
$x_9, x_{10}, x_{11}, x_{12}$	input IP packets received	1,2,3,4
$x_{13}, x_{14}, x_{15}, x_{16}$	output IP packets requested	1,2,3,4

Every input and output variable takes values 0 or 1. An input variable takes the value 1 when the corresponding observed variable is in the corresponding range. An output variables takes value 1 when the corresponding traffic category is observed.

The Feed-forward ANN has a one hidden layer with 4 hidden neurons. The activation function of input variables and output variables are linear. The activation function of the hidden neurons is logistic.

The next steps are: to train the model, to apply the model for intrusion prediction and to evaluate the model.

Conclusion

When creating data models for intrusion detection with neural network it is important to select the most suitable input variables – the ones which best describe the type of the network traffic. In the article we present a statistical approach for defining suitable input and output variables and to specify completely the neural network to use in intrusion detection.

References:

Al-Kasassbeh, Al-Naymat, Al-Hawari (2018) Using machine learning methodsfor detecting network anomalies within SNMP-MIB. Intenational Journal of Wireless and Mobile Computing, vol. 15, No. 1.

Bishop, M. (1995) Neural Networks for Pattern Recognittion. Oxford University Press. ISBN 0 19 853864 2 (Pbk)

Gotseva, D., R. Trifonov and P. Stoynov (2019) Neural Networks for Intrusion Detection.

Haddadi, F., A. Zincir-Heywood (2014) Benchmarking the Effect of Flow Exporters and Protocol Filters on Botnet Traffic Classification, IEEE Systems Journal 10 (4) (2016) 1390–1401. doi:10.1109/JSYST. 2014.2364743.

Ring, M., S. Wunderlich, D. Scheuring, D. Landes and A. Hotho (2019) A Survey of Network-based Intrusion Detection Data Sets. arXiv preprint: 1903:02460v2[cs.CR] 6 Jul 2019.

Sommer, R., V. Paxson (2010), Outside the Closed World: On Using Machine Learning For Network Intrusion Detection, in: IEEE Symposium on Security and Privacy, IEEE, 2010, pp. 305–316. doi:10.1109/SP.2010.25.

Stoynov, P. (2019a) Databases of anomalous traffic for estimation of Intrusion detection. Financial and Actuarial Mathematics-FAM-2019, PERUN-SPRINT, Sofia, Bulgaria, 2019, 3-8. ISSN 1314-460X.

Stoynov, P. (2019b) Evaluating Network Intrusion Detection Systems. - "Financial and Actuarial Mathematics-FAM-2019", PERUN-SPRINT., Sofia, Bulgaria, 2019, 9-16. ISSN 1314-460X.