Ninth International Conference
Globalization: business, finance and education – GB-2019

Neural Networks for Intrusion Detection

Daniela Gotseva
dgoceva@tu-sofia.bg
Roumen Trifonov
r_trifonov@tu-sofia.bg
Pavel Stoynov
todorov@feb.uni-sofia.bg

Reported also at the International Conference on High Technology for Sustainable Development - HiTECH-2019, 10-12 October 2019, Sofia, National Science and Technical Centre

Abstract.

This paper presents Intrusion Detection Systems (IDS), Intrusion Detection and Prevention Systems (IDPS) and their classification emphasizing on the use of neural networks in IDS.

Contemporary IDS usually include both signature verification and anomaly detection approaches realized by rule-based expert system and statistical module correspondingly. Neural networks may be used mainly as additional module to the statistical module to better recognize the user behavior. User behavior may be represented as frequency pattern of users command history. The paper presents an example of user profile vector for Unix-based platforms.

Key words: Neural Networks, Intrusion Detection Systems (IDS), Intrusion Detection and Prevention Systems (IDPS)

Intrusion Detection Systems and their classification

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 can be divided into two groups:

1. Host-based – for detecting hostile activity on host systems. They analyze information from operation systems and application files captured by software agents.

2. Network based – for detecting hostile activity on networks. They evaluate information from network communication, analyzing the stream of packets captured by sensors.

A typical IDPS consists of information capturing devices (sensors and agents), management server, database server and management console (Scarfone and Mell, 2007).

Sensors and scanners monitor the activity, capture and analyze security relevant information. The term sensor is typically used for network-based IDPS. The term agent is used in host-based IDPS.

A management server receives and analyzes the information gathered by agents and sensors (Trifonov et al., 2017).

IDPS perform data collection, data manipulation (feature selection and analysis) and preventive actions (Trifonov et al., 2017).

There are many IT security systems, which are not IDPS. Here are named some of them (Trifonov et al., 2017):

1. Cryptographic systems, mainly related to data and application security, like VPN, SSL, S/MIME, Kerberos, Radius etc.

2. Vulnerability assessment tools, related to application security, checking for vulnerabilities in operating systems and network services, for example Cybercop from Network Associates (NAI, 2019).

3. Network traffic monitoring systems, related to network security, used to detect Denial of Service (DoS) attacks.

4. Anti-virus systems, mainly related to host and network security, designed to detect malicious software, are very similar to IDPS, as viruses can be considered as simple, single, non-persistent intrusion.

The models for intrusion detection can be divided into two groups:

1. Signature verification. In this case, the system detects previously registered intrusion signature.

2. Anomaly detection. In this case, The system looks for abnormal behavior.

Most IDS commercial tools follow the signature verification detection system. However, they have some drawbacks (false alarms, signature of the attack can't be easily discovered, IDS should be periodically updated with new signatures).

Anomaly detection IDS are mainly in research stage today. Approaches for anomaly detection are: using statistical measures and suitable thresholds - predictive pattern generation (Teng et al. 1990) or state transition analysis (Porras et al. 1995); using rule-based expert systems (Garvey and Lunt, 1991); using soft computing Artificial Intelligence (AI) algorithms (neural networks, genetic algorithms etc.). The most common approach is the using of statistical measures combined with thresholds, but one serious limitation is to find the correct thresholds.

Contemporary IDS usually include both signature verification and anomaly detection realized by rule-based expert system and statistical module correspondingly.

Neural Networks for Intrusion Detection

The Artificial Neural networks (ANN) are information systems composed of multiple simple processing units (neurons) sharing their outputs with other neurons through weighted connections (Trifonov et al., 2017).

Recently, neural networks (Anderson, 1995) are applied to IDS (Fox et al. 1990; Debar et. al., 1992). They are commonly used in anomaly detection as alternative of the statistical and expert-system methods.

In anomaly detection technique, the main goal is to match an agent's observed behavior to his past behavior. The task is not easy, as agent's behavior can be very complex.

When statistical methods are used, false alarms can be generated due to invalid assumption about distribution of the audit data or missed detections can result from inability to distinguish intrusive from normal behavior on statistical basis.

To fix some gaps in statistical approach, an additional rule-based expert system is often used. With this expert system, an event can trigger a rule independently of the statistical classification of the event as normal or intrusive. But maintaining a complex rule-based system can be a challenge.

To fix the gaps of both statistical and expert-system approaches, the neural networks are considered as another component in anomaly-based IDS. The use of neural networks can overcome the following problems:

1. Neural networks don't require any accurate assumptions for statistical distribution of the subject behavior.

2. Neural networks may be used to evaluate the effectiveness of different sets of detection measures.

3. Neural networks are easier to modify than statistical or expert-system algorithms.

4. Neural networks can easy classify large number of users into groups based on their behavior, helping to overcome the difficulties with scaling.

The neural network itself however can't completely replace the IDS statistical component as the statistical approach explains which measures contribute to an event being considered anomalous. Currently it is not easy to find explanatory information from neural networks. Because of this neural networks are used in parallel with statistical and expert-system modules of IDS.

Two kinds of ANN are mainly applied in IDS: Multi-Level Feed-Forward ANN and Kohonen's Self-Organizing Maps (Vesely and Brechlerova, 2004).

Examples of use of Multi-Level Feed-Forward ANN and mainly Multi-Level Perceptrons (MLP) are the MLP used by Cannady and Mahaffey (1998) of Georgia Technical Research Institute for misuse detection, and MLP used by Cunningham and Lippmann (2000a, 2000b) of the MIT Lincoln Laboratory for searching attack-specific keywords in the network traffic.

Examples of use of SOM are the combinations MLP/SOM used by Cannady and Mahaffey (1998) and the SOM used by Girardin and Brodbeck (1998) and Girardin (1999) of the UBILAB laboratory to perform clustering of network traffic and detect attacks.

Many anomaly detection IDS are based on the general model proposed by Denning (1987). The model maintains a set of historical behavior profiles for users, matches an audit record with the appropriate profile, updates the profile whenever necessary, and reports normal behavior or anomalies detected.

Following this model Ryan et al. (1998) consider an anomaly based IDS that characterizes a legitimate user by the distribution of commands the user executes.

Building Neural network IDS module (NNIDS) for a particular computer system consists of the following three phases (Ryan et al., 1998):

1. Collecting training data. Obtain the audit logs for each user for a period of several days. For each day and user, form a vector that represents how often the user executed each of a set of commands. The number of

times the user executed each of these commands during the day is recorded, mapped into a nonlinear scale, and concatenated into an input vector, representing the usage pattern for that user for that day.

2. Training. Train the neural network to identify the user based on these command distribution vectors.

3. Performance. Let the network identify the user for each new command distribution vector. If the network's suggestion is different from the actual user, or if the network does not have a clear suggestion, signal an anomaly.

Ryan et al. (1998) built a NNID system and tested it on a machine that serves a particular research group at the Department of Electrical and Computer Engineering at the University of Texas at Austin. The machine has an operating system NetBSD which provides audit trail logging for accounting pur-poses and this option had been enabled on this system. The input layer consisted of 100 units, one for each command; the hidden layer had 30 units and the output layer 10 units, one for each user. The network was implemented in the PlaNet Neural Network simulator (Miyata, 1991).

An example of user profile vector for Unix-based platforms

In this article we propose design of a NNIDS which is modification of the design proposed by Ryan et al. (1998). The proposed design of NNIDS has the following features:

1. A user's profile vector with reduced dimension of 50 coordinates corresponding to 50 typical Unix commands is proposed. Ryan et al. (1998) propose similar vector of 100 coordinates. The list of selected here Unix commands is presented in Appendix 1.

2. To map the number of times the user executed a command during a day (24 hours) to user's profile coordinates, a non-liner scale of 5 intervals is used as in the Table 1. Ryan et al. (1998) use a non-linear scale of 11 intervals.

Table 1. Mapping of the number of times the user executed a command during a day

Number of use of a command per day	Mapped value
0-3	1
4-7	2
8-20	3
21-54	4
>55	5

3. One possible design of the ANN uses MLP and consists of 3 layers: input layer, hidden layer and output layer. Input layer contains 50 neurons (corresponding to the dimension of the user's profile). The hidden layer consists of 15 neurons and the output layer consists of 10 neurons (the number of the users). The time window is considered to be 20 days. Ryan et al. (1998) use a neural network with consisting of 3 layers with input and output layers having numbers of neurons equal to user vector dimension and number of users correspondingly, but 30 neurons in the hidden layer. Also or aim is to apply other ANN to select the best performers.

4. Another possible design of the ANN uses SOM based on Kohonen neural network (Kohonen, 2001). Here the task is to apply mapping of the user's profile to a SOM which is a grid with number of neurons equal to the number of users. A profile vector (called in SOM terminology also codebook vector) is assigned to every neuron, that will play the role of a typical user's profile. At the beginning, a subset of the data is randomly assigned to the neurons. During training, profiles are repeatedly presented – in random order – to the map. The neuron which is most similar to the current training profile, will be updated to become even more similar; a weighted average is used. Ryan et al. (1998) don't use Kohonen neural network.

5. The design is planned to be realized using R platform (Chambers, 1998). For MLP, the package *neuralnet* will be used and for SOM the package *kohonen* will be used. Ryan et al. (1998) realized their MLP model by PlaNet (Miyata, 1991) and don't apply SOM model.

Comparing generally the proposed here model and the model used by Ryan et al. (1998), we may say that the proposed model is more simplified and easy to apply (dimension of the user's profile vector is decreased two times, the number of the hidden neurons in the ANN is reduced two time). The proposed model uses a second option for the ANN – the Kohonen neural network.

Also, R platform offers more freedom for experimenting with different types of neural networks.

Summary

Contemporary IDS usually include both signature verification and anomaly detection approaches. ANN can be used mainly as additional module to the statistical and rule-based modules in anomaly detection part of the IDS.

User behavior may be represented as frequency pattern of users command history. Based on this idea, different models of NNIDS modules can be proposed, similarly to the proposed one in this article.

The study can be further developed in different directions:

1. Gathering data about user's profiles from different platforms.
2. Realizing the proposed ANN by R platform.
3. Training of the ANN and its use for user's profile mapping.
4. Building other NNIDS modules and using other applications of Artificial Intelligence to IDS to compare and select optimal variants.

References:

Anderson, J. (1995) An Introduction to Neural Networks. MIT Press.

Cannady, J., J. Mahaffey (1998) The application of Artificial Neural Networks to Misuse detection: initial results. Georgia Tech Research Institute. http://www.raid-symposium.org/raid98/Prog_RAID98/Talks.html#Cannady_34

Chambers, J. (1998) Programming with Data. Springer. New York.

Cunningham, R., R. Lippmann (2000a) Improving Intrusion Detection performance using Keyword selection and Neural Networks. MIT Lincoln University. http://www.ll.mit.edu/ IST/pubs. html

Cunningham, R., R. Lippmann (2000b) Detecting Computer Attackers: recognizing patterns of malicious,ste althy behavior -MIT Lincoln Laborator y - Presentation to CERIAS 11/29/2000. http://www.cerias.purdue.edu/secsem/abstracts0001.php

Debar, H., M. Becker, D. Siboni (1992). A neural network component for an intrusion detection system. In Proceedings of the 1992 IEEE Computer Society Symposium on Research in Computer Security and Privacy, 240–250.

Denning, D. (1987). An intrusion detection model. IEEE Transactions on Software Engineering, SE-13:222–232.

Fox, K., R. Henning, J. Reed, R. Simonian (1990). A neural network approach towards intrusion detection. In Proceedings of the 13th National Computer Security Conference, 125–134.

Garvey, T., T. Lunt (1991) Model-based intrusion detection. In Proceedings of the 14th National Computer Security Conference.

Girardin, L., D. Brodbeck (1998) A Visual Approach for Monitoring Logs. In Proceedings of the 12th System Administration Conference (LISA '98), p. 299-308, Boston, MA. http://www.ubilab.org/publications/index.html

Girardin, L. (1999) An eye on networ k intruder-administrator shootouts -UBS UBILAB In Proceedings of the 1st Workshop on Intrusion Detection and Network Monitoring, Santa Clara,CA. http://www.ubilab.org/publications/index.html

Kohonen, T. (2001) Self-Organizing Maps. Number 30 in Springer Series in Information Sciences. Springer-Verlag, Berlin, 3 edition.

Miyata, Y. (1991) A User's Guide to PlaNet Version 5.6 – A Tool for Constructing, Running, and Looking in to a PDP Network. Computer Science Department, University of Colorado, Boulder, Boulder, CO.

NAI (2019) Cybercop http://www.pgp.com

Porras, P., K. Ilgun, R. Kemmere (1995). State transition analysis: A rule-based intrusion detection approach. IEEE Transactions on Software Engineering,SE-21: 181–199.

Ryan, J., M.-J. Lin, R. Miikkulainen (1998) Intrusion Detection with Neural Networks. Advances in Neural Information Processing Systems, 10, Cambridge, MA, MIT Press.

Scarfone, K., P. Mell (2007) Guide to Intrusion Detection and Prevention systems (IDPS). National Institute of Standards and Technology.

Teng, H., K. Chen, S. Lu (1990). Adaptive real-time anomaly detection using in-ductively generated sequential patterns. In Proceedings of the 1990 IEEE Symposium on Research in Computer Security and Privacy, 278–284.

Trifonov, R., G. Tsochev, R. Yoshinov, S. Manolov, G. Pavlova (2017) Conceptual model for cyber intelligence network security system. International Journal of Computers. Vol. 11, 2017.

Vesely, A., D. Brechlerova (2004) Neural networks in intrusion detection systems.Agric. econ. – Czech, vol. 50, pp. 35-39.

Appendix 1

50 Unix commands used in user's profile vector
(Explanation of commands is taken from the Help system of Unix)

1. as - a generic name for an assembler on Unix. On many systems the standard or pre-tor installed assembler is the GNU Assembler, commonly called GAS, whose executable is simply named as.

2. cat - cat (short for "concatenate") command is one of the most frequently used command in Linux/Unix like operating systems. cat command allows us to create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

3. cp - cp is a command in various Unix and Unix-like operating systems for copying files and directories.

4 . cpp - the C preprocessor, often known as cpp, is a macro processor that is used automatically by the C compiler to transform your program before compilation. It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.

5. cvs - Concurrent Versions System command - a tool for version control.

6. df - shows information about the file system on which each FILE resides, or all file systems by default.

7. du - summarizes disk usage of each FILE, recursively for directories.

8. egrep - grep or Global Regular Expression Print is the main search program on Unix-like systems which can search for any type of string on any file or list of files or even output of any command. It uses Basic Regular Expressions apart from normal strings as a search pattern. egrep is same as 'grep -E' or 'grep –extended-regex', which uses extended regular expression.

9. elm - elm is a text-based email client commonly found on Unix systems. First released in 1986, it became popular as one of the first email clients to use a text user interface, and as a utility with freely available source code. The name elm originated from the phrase ELectronic Mail.

10. ftp - ftp is the user interface to the Internet standard File Transfer Protocol. FTP is the simplest file transfer protocol to exchange files to and from a remote computer or network.

11. gcc - the GNU Compiler Collection (GCC) is a compiler system produced by the GNU Project supporting various programming languages. GCC is a key component of the GNU tool chain and the standard compiler for most Unix-like operating systems. The Unix command for compiling C code is gcc. This is a compiler from Gnu for Linux. GCC was originally written as the compiler for the GNU operating system. Suppose you have a file hello.c which you want to compile (or use hello.c.) Type the following on the command line: gcc hello.c.

12. gdb - gdb is a debugger for C (and C++). It allows you to do things like run the program up to a certain point then stop and print out the values of certain variables at that point, or step through the program one line at a time and print out the values of each variable after executing each line.

13. ghostview - Ghostscript, Ghostview and GSview. This was the home page for Ghostscript, an interpreter for the PostScript language and for PDF. Ghostscript is a suite of software based on an interpreter for Adobe Systems' PostScript. Ghostscript has been ported to many operating systems, including Unix-like systems, classic Mac OS, OpenVMS, Microsoft Windows.

14. gmake -gmake is the FSF's implementation of make. GNU Make has many powerful macro features for use in makefiles, beyond what other Make versions have/

15. grep – searches for patterns in files or standard input. Pattern is, by default, a basic regular expression (BRE). Example: grep -i 'hello world' menu.h main.c

16. gzip - Gzip(GNU zip) is a compress tool which is available in most of the Linux/Unix based operating systems. Until recent years gzip and bzip2 are most commonly used data compression tools in Linux/Unix. Though gzip compress ratios are not good when compared to bzip2 but it is popular among masses.

17. hostname -hostname is used to display the system's DNS name, and to display or set its hostname or NIS (Network Information Services) domain name. The host name is usually set once at system startup in the script /etc/init.d/hostname.sh normally by reading the contents of a file which contains the host.

18. id - id command is one of the basic Unix commands, confirming the identity of a specified Unix user. Unix-like operating systems identify a user within the kernel by a value called a user identifier, often abbreviated to user ID or UID. The UID, along with the group identifier (GID) and other access control criteria, is used to determine which system resources a user can access.

19. ld - the ld command, also called the linkage editor or binder, combines object files, archives, and import files into one output object file, resolving external references. It produces an executable object file that can be run.

20. lpq - lpq shows the current print queue status on the named printer. Jobs queued on the default destination will be shown if no printer or class is specified.

21. ls - ls command in Linux/Unix. ls is a Linux shell command that lists directory contents of files and directories.

22. mail - the Mail command in unix or linux system is used to send emails to the users, to read the received emails, to delete the emails etc.

23. make - the purpose of the make utility is to determine automatically which pieces of a large program need to be recompiled, and issue the commands to recompile them. In fact, make is not limited to programs.

24. man - with the man command, you can retrieve the information in the manual and display it as text output on your screen.

25. mkdir - create directories, if they do not already exist.

26. more - more is a command to view (but not modify) the contents of a text file one screen at a time. It is available on Unix and Unix-like systems, DOS, FlexOS, OS/2, Microsoft Windows and ReactOS. Exit with 'q'.

27. mv - the mv command is a command line utility that moves files or directories from one place to another. It supports moving single files, multiple files and directories.

27. netstat (network statistics) – it is a command line tool for monitoring network connections both incoming and outgoing as well as viewing routing tables, interface statistics etc. It is available on all Unix-like Operating Systems and also available on Windows OS as well. It is very useful in terms of network troubleshooting and performance measurement and it is one of the most basic network service debugging tools, telling you what ports are open and whether any programs are listening on ports.

28. ping - the ping command sends ICMP ECHO_REQUEST packets to network hosts and reports on the response from the remote server, outputting to standard output. It can be used to check if a remote host is up, or that network interfaces can be reached.

29. ps - in most Unix-like operating systems, the ps program (short for "process status") displays the currently-running processes. A related Unix utility named top provides a real-time view of the running processes.

30. pwd - the pwd command is a command line utility for printing the current working directory. It will print the full system path of the current working directory to standard output. The pwd command is normally a shell builtin meaning it is part of the code that runs the shell rather than an external executable.

31. rcp - the rcp command is used to copy files between different computers without starting an FTP session or logging into the remote system explicitly. To use the rcp command, you must have every system you intend to use as a source or destination in your .rhosts file

32. rm - rm (Unix) In computing, rm (short for remove) is a basic command on Unix and Unix-like operating systems used to remove objects such as computer files, directories and symbolic links from file systems and also special files such as device nodes, pipes and sockets.

33. rsh - rsh is used to execute commands on a remote machine. The rsh command executes the command or a program in another host from current working machine without having to login into that remote machine by entering a password as in ssh. You can run any unix command, or a shell script of a remote host.

34. sed - this command in UNIX is stands for stream editor and it can perform lot's of function on file like, searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace.

35. sendmail - a general purpose internetwork email routing facility that supports many kinds of mail-transfer and delivery methods, including the Simple Mail Transfer Protocol (SMTP) used for email transport over the Internet.

36. sh - sh is a command language interpreter that executes commands read from a command line string, the standard input, or a specified file. The Bourne shell was developed in 1977 by Stephen Bourne at AT&T's Bell Labs in 1977. It was the default shell of Unix Version 7.

37. sort - sorts the contents of a text file, line by line. It is a standard command line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order.The sort command is a command line utility for sorting lines of text files.

38. strip - In Unix and Unix-like operating systems, the strip program removes inessential information from executable binary programs and object files, thus potentially resulting in better performance and sometimes significantly less disk space usage ('inessential information' means information that is not required for correct functioning of the binary in normal execution).

39. stty - stty command is used to manipulate the terminal settings. You can view and modify the terminal settings using this command.

40. tail - tail has two special command line option -f and -F (follow) that allows a file to be monitored. Instead of just displaying the last few lines and exiting, tail displays the lines and then monitors the file. As new lines are added to the file by another process, tail updates the display.

41. tar - in Unix, the name of the tar command is short for tape archiving, the storing of entire file systems onto magnetic tape, which is one use for the command. However, a more common use for tar is to simply combine a few files into a single file, for easy storage and distribution.

42. tcsh - a Unix shell based on and compatible with the C shell (csh). It is essentially the C shell with programmable command-line completion, command-line editing, and a few other features.

43. top - top command displays processor activity of your Linux box and also displays tasks managed by kernel in real-time. It'll show processor and memory are being used and other information like running processes. This may help you to take correct action. Exit with 'q'.

44. tput - in computing, tput is a standard Unix operating system command which makes use of terminal capabilities.Depending on the system, tput uses the terminfo or termcap database, as well as looking into the environment for the terminal type. Tput was provided in UNIX System V in the early 1980s.

45. tr - the tr command in UNIX is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace. It can be used with UNIX pipes to support more complex translation.

46. tty - "tty" originally meant "teletype" and "pty" means "pseudo-teletype". In UNIX, /dev/tty* is any device that acts like a "teletype", i.e., a terminal. Called teletype because that's what we had for terminals in those benighted days.

47. vi - the vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems.

48. w - the command w on many Unix-like operating systems provides a quick summary of every user logged into a computer, what each user is currently doing, and what load all the activity is imposing on the computer itself. The command is a one-command combination of several other Unix programs: who, uptime, and ps -a.

49. wc - wc (short for word count) is a command in Unix-like operating systems. The program reads either standard input or a list of files and generates one or more of the following statistics: newline count, word count, and byte count. Newer versions of wc can differentiate between byte and character count.

50. whereis - finds the binary, source code and man page for specified program or command.

The length n and the content of the user profile vector may be modified according to the specific Unix versions and specific host activities. Another important parameter is the size of the observing window m. It is the length of the observations vector for determining the frequencies of the used commands. The network receives the m most recent commands as its input. In the case when network is recurrent, part of the output is fed back as the input for the next step; thus, the network is constantly observing the new trend and “forgets” old behavior over time. If m is too small, there will be many false positives; if it is too big, the network may not generalize well to novel sequences.