Usage of Holt-Winters Model and Multilayer Perceptron in Network Traffic Modelling and Anomaly Detection

Maciej Szmit
Orange Labs Poland, 7 Obrzeżna Street, 02-691 Warsaw, Poland
E-mail: maciej.szmit@gmail.com, http://maciej.szmit.info

Anna Szmit
Technical University of Lodz, Department of Management, 266 Piotrkowska Street, 90-924 Lodz, Poland
E-mail: agorecka@p.lodz.pl, http://anna.szmit.info

Sławomir Adamus
Technical University of Lodz, Computer Engineering Department, 18/22 Stefanowskiego Street, 90-924 Lodz, Poland
AMG.lab, 11 Lakowa Street, 90-562 Lodz, Poland
E-mail: slawomir.adamus@hotmail.com

Sebastian Bugała
Technical University of Lodz, Computer Engineering Department, 18/22 Stefanowskiego Street, 90-924 Lodz, Poland
E-mail: sebastian.bugala@hotmail.com

Keywords: network behavioral anomaly detection, Holt-Winters model, multilayer perceptron

Received: September 16, 2012

This paper presents results of analysis of few kinds of network traffic using Holt-Winters methods and Multilayer Perceptron. It also presents Anomaly Detection – a Snort-based network traffic monitoring tool which implements a few models of traffic prediction.

Povzetek: Predstavljena je metoda za modeliranje in iskanje anomalij v omrežju.

1 Introduction

In modern computer networks and high-loaded business or industrial systems there is a need of continuous availability of services and hosts (see e.g. [28], [29] [30] [34]). Inaccessibility of some mission critical can cause large impact to business processing continuity and this as a result would generate looses. Solution for such potential problems could be permanent and uninterrupted supervision on network health. This in turn can be achieved by implementation of some monitoring solution. Efficient monitoring method helps achieve high service availability and it will be a good idea to extend network security by tools such as Intrusion Detection System, Intrusion Prevention System and Unified Thread Managers (see e.g. [32] [33]). IDS is a tool which monitors and analyses in real time every aspect of inbound and outbound traffic of the network. Based on the analysis and based on one of the mechanisms responsible for threat detection creates reports of the abnormalities of network traffic. Most common mechanisms which detect threats used in IDS are misuse detection and anomaly detection, they are two different approaches to threat detection, first one relays on determination abnormal parameters and network traffic behavior, everything which we do not know is treated as normal, second one is a reverse of the first one, it treats everything which deviates from the standard is treated as potential threat. IDS on its own only reports and logs the abnormalities and does not take any further actions and his role is to report to administrator which is whom decides what action should be taken to prevent imminent danger which can be a cumbersome for the administrator with a large number of notifications. In order to relieve the amount of work of administrator, ideas of IDS have been extended by possibility to take defined actions immediately in case of detection of typical and schematic threats for the network, as a result IPS was created which is a variety of IDS which is compatible with tools such as firewalls and control its settings in order to counter the threat.

A typical representative of the above-described tool is Snort (see e.g. [2] [3] [31]), a software type of IDS/IPS based on mechanism which detects attack signatures originally intended only for the Unix platform, but now also transferred to the Windows operating system, developed on the principles of open source software licenses. Large capacity and performance are characteristics that gained snort popularity among users. Its modular design makes the software very flexible and thus can be easily adapted to the requirements of the currently analyzed network environments, and expand its functionality.

This article extends demonstration of the capabilities of the AnomalyDetection tool (basic overview of the tool was published in [15] and [36]) created for network

monitoring and future network traffic forecasting Snort-based applications using the flexibility and easy extensibility (the ability to create own preprocessors and postprocessors) of this program. The preprocessor was developed to extends Snorts possibilities of network traffic analysis by anomaly detection mechanism [4]. Combination of the two mechanisms (i.e., misuse detection and anomaly detection) provides more comprehensive protection against all types of threats, even those partially abstract, such as the malice of employees. Tools included in the Anomaly Detection 3.0 allows analysis of movement, its forecasting with help of its advanced statistical algorithms, evaluation of created forecasts, real-time monitoring and verifying that the individual volumes of network traffic parameters do not exceed the forecasted value and in case of exceeding the norms to generate the appropriate messages for the administrator who should check each alarm for potential threats.

Current (3.0) version (see e.g. [5], [6]) of AnomalyDetection provides monitoring of following network traffic parameters: total number of TCP, UDP, and ICMP packets, number of outgoing TCP, UDP, and ICMP packets, number of incoming TCP, UDP, and ICMP packets, number of TCP, UDP, and ICMP packets from current subnet, number of TCP packets with SYN/ACK flags, number of outgoing and incoming WWW packets – TCP on port 80, number of outgoing and incoming DNS packets – UDP outgoing on port 53, number of ARP-request and ARP-reply packets, number of non TCP/IP stacks packets, total number of packets, TCP, WWW, UDP, and DNS upload and download speed [kBps].

Whole Anomaly Detection application consists of three parts: Snorts preprocessor, Profile Generator and Profile Evaluator. Data exchange between these parts is realized by CSV (Comma Separated Values) files (see: Figure 1).

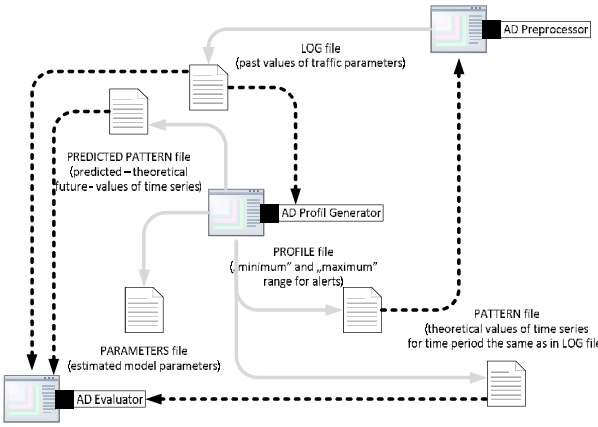

Figure 1: Anomaly Detection data flow diagram. Source: [15].

Gray solid arrows means saving to file and black dotted – reading from file. Particular files stands for:

- Log file – this file gathers all network traffic data collected with AD Snort preprocessor. Data from this file is next used by Profile Generator for network traffic forecasting.
- Profile file – this file stores network profile computed with Profile Generator. This file is generated by Profile Generator and used by AD preprocessor for detecting anomalies and generating alerts. After every passed time period preprocessor reads profile file and looks for data corresponding to current period. If value for some counter exceeds minimum (MIN) to maximum (MAX) range then alert is generated.
- Predicted pattern file – predicted pattern file contains predicted future data for network – in fact this is the same file as profile file, but with single value for each counter. This is necessary for evaluating profile in AD Evaluator script. Structure of pattern file is the same as log file.
- Pattern file – this file is created like predicted pattern file, but network traffic profile stored in this file is historical data.
- Parameters file – this file stores information for method of profile generation and method parameters values. This file has different structure for every algorithm of profile generation.
- Structures of log and profile files can be found in [15]. Anomaly Detection have two main modes:
- data acquisition mode – only network traffic statistics are saved into log file. Only log file is created in this mode.
- alerting mode – instead of data acquisition there is also created profile file and current traffic statistics are compared to values stored in profile file. In this mode log and profile file are required.

Pattern, predicted pattern and parameters files are always optional and they're useful for future research.

Anomaly Detection 3.0 can be downloaded from http://anomalydetection.info [24]. Preprocessor is available as source or RPM package. Both Profile Generator and Evaluator are available as R scripts – additional R CRAN (free) software is required for use R scripts. Additional instalation, update and removal scripts are provided for Profile Generator and Evaluator.

2 Preprocessor

The main part of the Anomaly Detection system is a preprocessor written in C programming language, designed to enhance Snort possibilities to monitor, analyze and detect network traffic anomalies using NBAD (Network Behavioral Anomaly Detection) approach. The first version of AnomalyDetection preprocessor [6] for Snort version 2.4x was published in a Master's Thesis [25] in 2006. Next the project has been developed (see e.g. [5] [7] [8] [9] [17]) till the current version 3.0 designed for Snort 2.9.x.

The main task of the preprocessor is anomaly detection, realized by using a simple algorithm based on data acquisition and subsequent comparison of the collected values with pattern. Preprocessor reads a predicted pattern of the network traffic (of all

parameters) from the 'profile' file and generates alert when the current value exceeds 'minimum' to 'maximum' range for the current moment (the moment is given by day of the week, hour, minute and second corresponding to the intervals from the log file) from the profile file.

The profile can be generated 'manually', using external tools, or by a Profile Generator using appropriate model, based on historic values from the log file. The architecture affords easy implementation of different statistical models of the traffic and usage of different tools (i.e. statistical packets) for building profiles. Data from the profile is read in intervals defined by the user, there is only one line read into the structure at a time, this gives possibility to dynamically alter the profile file. In case of failure to find the correct entry in the profile, anomaly report module is automatically disabled to prevent generation of false positive alerts.

As mentioned above the current version of the preprocessor can work with adaptive network models through changes in the algorithm which loads profile information. Abandoned single network profile load for the load of single-line in specified time interval. Profile data is loaded at exact time of writing counter to the log file. This solution although increases the number of I/O operations adversely affecting the performance but also supports replacing another model during runtime without having to restart whole application. In addition, all the calculations have been relegated to third-party applications and the profile has been changed so that it contains the minimum and maximum value. This approach makes the preprocessor is more flexible and efficient, does not limit the user to use a single method to generate a network profile, the profile can be freely generated by any application while maintaining only the appropriate input format. Reporting anomalies was adjusted to snort standards by implementing a mechanism which reports events and handle these events by dedicated preprocessor rules. The user can freely adjust the rules to fit his needs, for example; the content of messages stored in the log, which is a priority or which action should be taken when matching rules. These changes make the application more customizable and user-friendly. Improving algorithm for packet acquisition by removing unnecessary comparisons and optimizations of other ones and increased capacity of counters made it possible to use preprocessor in networks with high bandwidth 1Gb and above.

The next function of the preprocessor is generating alerts. Preprocessor reads a predicted pattern of the network traffic (of all parameters) from the 'profile' file and generates alert when the current value exceeds 'minimum' to 'maximum' range for the current moment (the moment is given by day of the week, hour, minute and second corresponding to the intervals from the log file) from the profile file.

The profile can be generated 'manually', using external tools, or by a Profile Generator using appropriate model, based on historic values from the log file. The architecture affords easy implementation of different statistical models of the traffic and usage of different tools (i.e. statistical packets) for building profiles. Data from the profile is read in intervals defined by the user, there is only one line read into the structure at a time, this gives possibility to dynamically alter the profile file. In case of failure to find the correct entry in the profile, anomaly report module is automatically disabled to prevent generation of false positive alerts.

3 Profile Generator

In previous versions of AnomalyDetection system profile generation module was included in preprocesor module – because of this whole application was inflexible. The current version of Profile Generator (see e.g. [7] [8] [9]) have been separated into independent module which can be used to compute statistical models not only for AD preprocessor. Furthermore current version is based on R language / environment (The R Project for Statistical Computing) (see e.g. [10] [11] [12] [13] [14]) which is more flexible and user-friendly than previous implementation in C language. R-project is an free, open source packet for statistical computing and graphics. In this implementation optional packages for R: tseries, quadprog, zoo and getopt are used.

The whole implementation of Profile Generator is divided into few parts. First part prepares data from log file for further calculations and other parts – depending on the given parameters – calculates future network traffic forecasts. At the end all computed values are written into proper files – based on given runtime parameters. Data flow in ProfileGenerator module is shown on Figure 2.

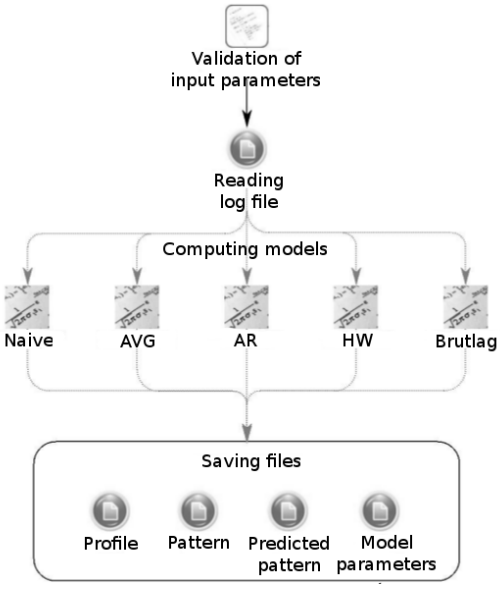

Figure 2: Profile Generator data flow diagram. Source: [35].

Profile Generator is controlled with parameters passed for script execution – all script parameters are handled with `getopt()` function.

Particular columns of specification matrix contains respectively:

- long flag name
- short flag

- parameters arguments
- arguments type
- description

Profile Generator actually implements five methods of profile file generation: moving average, naive method, autoregressive time series model, Holt-Winters model and Brutlags version of HW model (see e.g. [1] [17]). The value of dependent variable is given as follows:
Moving average:

$$\hat{y}_t = \frac{\sum_{i=t-k}^{t-1} y_i}{k} \quad (1)$$

Naive method:

$$\hat{y}_t = y_{t-T} \quad (2)$$

where T is day or week period, or

$$\hat{y}_t = y_{t-1} \quad (3)$$

Autoregressive time series model:

$$\hat{y}_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + ... + a_k y_{t-k} \quad (4)$$

Holt-Winters model:

$$\hat{y}_t = L_{t-1} + P_{t-1} + S_{t-T} \quad (5)$$

where:

L is level component given by:

$$L_t = \alpha(y_t - S_{t-T}) + (1-\alpha)(L_{t-1} + P_{t-1}) \quad (6)$$

P is trend component given by:

$$P_t = \beta(L_t - L_{t-1}) + (1-\beta)P_{t-1} \quad (7)$$

S is seasonal component given by:

$$S_t = \gamma(y_t - L_t) + (1-\gamma)S_{t-T} \quad (8)$$

Brutlag method:

$$\hat{y}_t^{\max} = L_{t-1} + P_{t-1} + S_{t-T} + md_{t-T} \quad (9)$$

$$\hat{y}_t^{\min} = L_{t-1} + P_{t-1} + S_{t-T} - md_{t-T} \quad (10)$$

where:

L, P and S are the same as in Holt-Winters model

d is predicted deviation given by:

$$d = \gamma|y_t - \hat{y}_t| + (1-\gamma)d_{t-1} \quad (11)$$

where:

k is number of measurements in time series

t is moment in time

$\hat{y}$ is predicted value of variable in moment t

y_t is real (measured) value of variable in moment t

T is time series period

α is data smoothing factor

β is trend smoothing factor

γ is the seasonal change smoothing factor

m is the scaling factor for Brutlags confidence bands

4 Implementation of Naïve Method

Naïve method is the simplest method implemented in Profile Generator module. For computing profile with this method PG must be launched with '-m NAIVE' parameter. Additional '--naive' parameter can be used for defining detailed method 'periodicity'. Method implement three version of naïve prediction – LAST, DAILY and WEEKLY. For LAST version forecasted data are defined as the same as previous measurement. DAILY version means that predicted values for some day would be the same as values in previous day of given time-series. The last version stand for algorithm in which forecasted values are determined based on logged data for the same day-of-week in previous week.

Because of simplicity if this method it should be used only in adaptive startup mode – this will cause less false-positive alerts and more dynamically prediction. In this mode profile is recalculated in regular intervals of time, so predicted values refreshes with every oncoming period of counter values registration. Figure 3 shows graph with predicted values with 5 period interval of method recalculation. It can be observed step changes of predicted values in succeeding periods.

Y-axis on Fig 5 stands for minimal and maximal border of permitted values for total number of TCP packets. X-axis stands for sample number in forecasted time-series

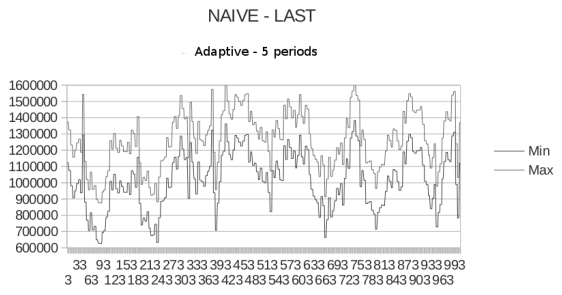

Figure 3: Naive method running in adaptive mode with 5 period interval of recalculation. Source: [35].

5 Implementation of Moving Average Method

Moving average method is computed when Profile Generator is run with '-m AVG' parameter set. Detailed method periodicity and length of the horizon of values used for calculation can be defined with '--avg' parameter.

Similar to the naïve method – there are three versions of periodicity: LAST, DAILY and WEEKLY.

There is also required second parameter which stands for number of values used to compute moving average. For example 'DAILY,3' means that values from three previous days would be used to compute prediction, 'LAST,5' means that average would be computed using five previous values registered in log file.

6 Implementation of Autoregressive Model

AR model can be calculated when run with '-m AR' parameter. Calculations in this method are based on ar() function from package stats in R environment. Function ar() fits an autoregressive time series model to given data and it is wrapper for the functions: ar.yw, ar.burg, ar.ols and ar.mle. Setting 'method' parameter to ar() function defines the method used to fit the model.

There are available four algorithms used to fit model to given time-series: Yule-Walkers, Burgs, MLE (maximum likelihood) and OLS (ordinary least squares).

7 Implementation of Holt-Winters Model

The Holt-Winters model, called also the triple exponential smoothing model, is a well-known adaptive model used to modeling time series characterized by trend and seasonality (see e.g. [20], [19] p. 248, [18], [21], [22]). The model is sometimes used to modeling and prediction of network traffic (see e.g. [23],[7], [8]).

For computing an Holt-Winters model Profile Generator must be launched with parameter '-m HW'. Optional parameter '--hw' can be set for defining model periodicity and subset of data used to build model.

Implementation of Holt-Winters prediction method in Profile Generator is based on function `HoltWinters()` from package stats. `HoltWinters()` functions requires time series data as object of class 'ts' (time-series object). Object 'ts' is created as follows:

```
ts_obj<-
ts(log.data[,column.log], frequency=pr
ofile.config.frequency, start=c(as.num
eric(log.first.date),log.first.sample.
no))
```

Function 'ts' gets in this implementation 3 parameters:

- data – a numeric vector of the observed time-series values
- frequency – the number of observations per unit of time
- start – the number of observations per unit of time. This parameter can be a single number or a vector of two integers – because of this in our implementation human-readable date from log file is converted into numeric value and second value is number of sample of first observation in the day.

Next `HoltWinters()` function computes Holt-Winters filtering of a given time series. Function tries to find the optimal values of α or β or γ by minimizing the squared one-step prediction error with `optim()` function. Start values for L, P and S are inferred by performing a simple decomposition in trend and seasonal component using moving averages – it is realized with `decompose()` function.

Figure 4 shows one weekly period (from January 1st to January 7th) of testing data.

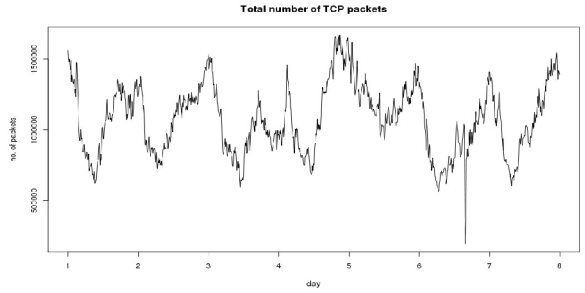

Figure 4: One period of testing data. Source: own research.

`Decompose()` function decomposes a time series into seasonal, trend and irregular components using moving averages. For testing data `decompose()` function returns values with trend, seasonal and random component. Figure 5 shows those decomposed data.

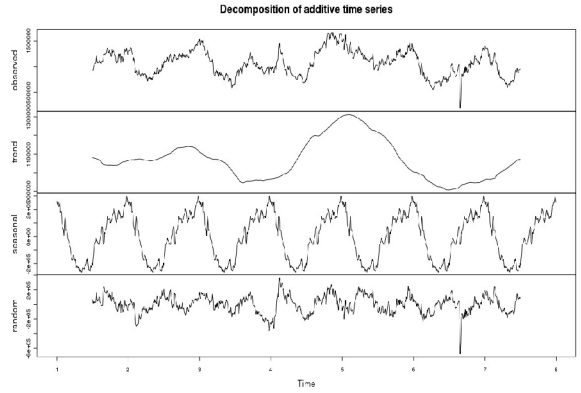

Figure 5: Decomposed time series. Source: own research.

`HoltWinters()` function estimates HW model smoothing parameters (alpha, beta and gamma), which were for testing data as follows (see: Figure 6). Figure 7 shows Holt-Winters fitted to observed comparison.

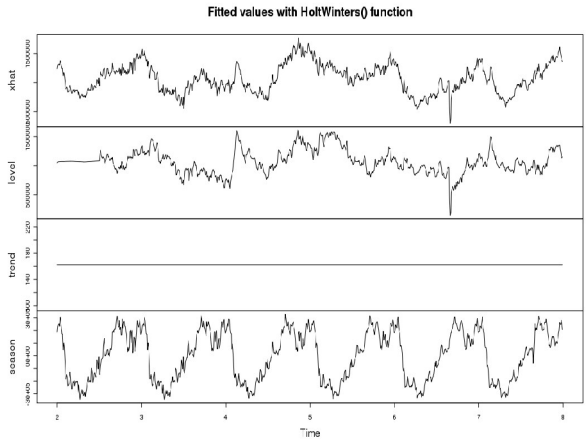

Figure 6: Fitted Holt-Winters. Alpha=0.8140128; beta=0; gamma=1. Source: own research.

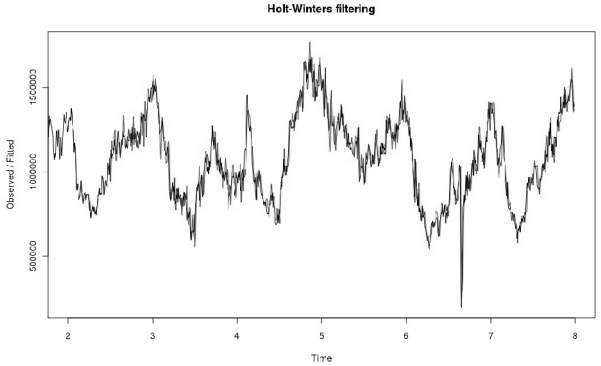

Figure 7: Holt-Winters fitted to observed comparison. Source: own research.

Fitted values compared to observed values for given testing data:

Black line stands for observed data and gray line stands for fitted model (in most range black line covers gray).

When Holt-Winters model is computed, then future prediction can be calculated simple with `predict.HoltWinters()` function. `Predict()` function takes in this case two arguments:

HoltWinters object with fitted model parameters

number of future periods to predict

Function returns a time series of the predicted values for given future periods. For testing data values returned from `predict()` function are shown on Figure 8.

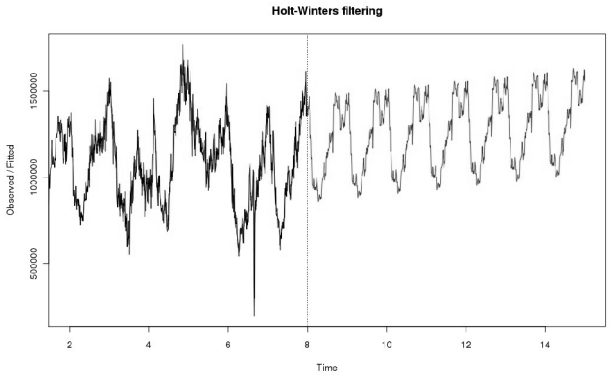

Figure 8: Holt-Winters prediction. Source: own research.

8 Brutlags Algorihm

Holt-Winters method was used to detect network traffic anomalies as described in the article [1]. In that paper, the concept of “confidence bands” was introduced. As described in the article, confidence bands measure deviation for each time point in the seasonal cycle and this mechanism bases on expected seasonal variability.

Illustration Fig 9 shows computed confidence bands for HW time series prediction.

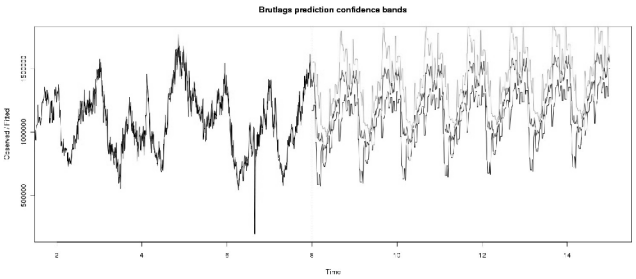

Figure 9: Brutlags confidence bands. Source: own research.

Confidence band is computed by comparing last period of collected network traffic values with fitted Holt-Winters values for the same period. Subtract of real and predicted values is next scaled with γ estimated by Holt-Winters function – obtained value is finally multiplied by scaling factor. Confidence band width is controlled with '--scale' parameter – above example is computed with scale parameter value of '2'. Brutlag proposes sensible values of '--scale' parameter are between 2 and 3. Particular lines stands for:

- black – observed values of time series
- medium gray – computed prediction of time series with Holt-Winters model
- light gray – upper bound of Brutlags confidence band
- gray – lower bound of Brutlags confidence band

9 Usage of Profile Generator

Generator can be launched like any script in CLI (Command Line Interface) of operating system with R software and necessary packages installed. Scripts available at [24] were tested on few GNU / Linux distributions: Fedora, Oracle Linux, CentOS, Debian, and Ubuntu. Parameters for Profile Generator script are validated against bellow BNF notation grammar:

```
  ad_profilegenerator.r <mode>
  <mode>       ::= <m_help> | <m_generate>
  <m_help>     ::= -(-help|h)
  <m_generate> ::=  <log>  <profile>  <evaluator>
<pattern> <model_param> <method> <ahead> <scale>
<verbose>
  <log>        ::= -(-log|l) <<log_file_path>>
  <profile>    ::= -(-profile|p)
<<profile_file_path>> | <<empty>>
  <evaluator>  ::= -(-evaluator|e)
<<predicted_pattern_file_path>> | <<empty>>
  <pattern>    ::= -(-pattern|P)
<<pattern_file_path>> | <<empty>>
  <model_param>       ::= -(-save|s)
<<model_parameters_file_path>> | <<empty>>
  <verbose>    ::= -(-verbose|v) | <<empty>>
  <ahead>             ::= -(-ahead|a)
<ahead_val> | <<empty>>
  <ahead_val>  ::= WEEK|MONTH|<number>
  <scale>             ::=           -(-scale|d)
<<scale_parameter>> | <<empty>>
  <method>     ::= -(-method|m)  <pred_method>  |
<<empty>>
  <pred_method>       ::=  AVG  <avg_param>  |
NAIVE  <naive_param>  |  AR  <ar_param>  |  HW
<hw_param> | BRUTLAG <brutlag_param>
  <avg_param>  ::= --avg <avg_value> | <<empty>>
```

```
  <naive_param>       ::= --naive <naive_value>
| <<empty>>
  <ar_param>   ::= --ar <ar_value> | <<empty>>
  <hw_param>   ::= --hw <hw_value> | <<empty>>
  <brutlag_param>::=  --brutlag  <brutlag_value>
| <<empty>>
  <avg_value>  ::= (LAST|DAILY|WEEKLY),<number>
  <naive_value>::= (LAST|DAILY|WEEKLY)
  <ar_value>   ::=
(DAILY|WEEKLY),(YW|BURG|MLE|OLE)
  <hw_value>   ::= (DAILY|WEEKLY)
  <brutlag_value> ::= (DAILY|WEEKLY)
  <number>     ::=
<number><number>|0|1|2|3|4|5|6|7|8|9
```

Sense of each parameter impact is clarified under '--help' parameter. At least one of <profile>,<evaluator>,<pattern>, or <model_param> parameter should be set for any sense of running script.

For example the simplest naïve prediction for real data stored in 'log.csv' file with saving profile data to 'profile.csv' file can be launched with:

```
./ad_profilegenerator.r -l log.csv -p
profile.csv -m NAIVE --naive LAST
```

Prediction for one week for the same file based on Holt-Winters algorithm with daily periodicity and with 'verbose' mode can be calculated with:

```
./ad_profilegenerator.r -l log.csv -p
profile.csv -m HW --hw DAILY –ahead
WEEK -v
```

10 Evaluator

Profile Evaluator is` the third part of Anomaly Detection project. This script is designed for fast evaluation of profile file compared to log file. This script calculates simple statistic $\frac{MAE}{M}$ for two files. Main application of Evaluator is to check fit between pattern and current logged values (with log and pattern file) or between model and historical data (log and predicted pattern file).

MAE means Mean Absolute Error and M means Mean.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\left|y_t - \hat{y}_t\right| = \frac{1}{n}\sum_{t=1}^{n}\left|e_t\right| \quad (12)$$

$$M = \frac{1}{n}\sum_{t=1}^{n} y_t \quad (13)$$

where:

y_t is real (current) value of counter in moment t

$\hat{y}_t$ is predicted (estimated) value of counter in moment t

e_t is prediction error in moment t

Calculated values for each counter can be stored in output file when '-s' parameter is set. Exemplary comparison of real registered values with its prediction is shown on Fig 10.

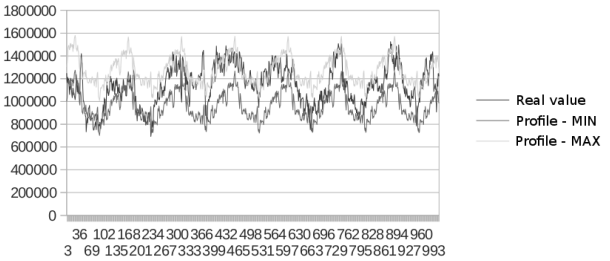

Figure 10: Real values compared to AVG - DAILY,3 prediction. Source: [35].

Profile Evaluator script is launched likewise Profile Generator script. Profile Evaluator script parameters grammar looks as follows:

```
  ad_evaluator.r <mode>
  <mode>              ::= <m_help> |
<m_evaluate>
  <m_help>            ::= -(-help|h)
  <m_evaluate> ::= <log> <pattern> <save> <skip>
<verbose>
  <log>         ::= -(-log|l) <<log_file_path>>
  <pattern>::= -(-pattern|p)
<<pattern_file_path>>
  <save>                      ::= -(-save|s)
<<save_maem_file_path>>  <<empty>>
  <skip>              ::= -(-skip|S) <number> |
<<empty>>
  <verbose>    ::= -(-verbose|v) | <<empty>>
```

Evaluation of pattern stored in 'pattern.csv' file compared with log data stored in 'log.csv' file can be done with:

```
./ad_evaluator.r -l log.csv -p
pattern.csv --verbose
```

11 Multilayer Perceptron

All our previous models can be classified as statistical model assigned to one of two groups: Time Series Models and descriptive models. The next step is usage of artificial-intelligence methods, particularly Artificial Neural Networks (ANN) which are implemented only as offline models in the current state of our research.

Artificial Neural Networks are the mathematical models inspired by biological neural networks. ANN consist of an interconnected group of artificial neurons operating in parallel. ANN function is determined by the weights of the connections between neurons, which usually change during a learning phase. There are a lot of types and architectures of ANN according on their purpose.

Because of the nature of IDS there are two main groups of issues: pattern recognition, especially classification and prediction. These issues correspond with two main areas of application of ANN. In consequence ANN can be used for intrusion detection in two main ways: as a classifier which determine whether a given object (for example: network packet, e-mail, network flow) is normal or suspicious and as a predictor which try to forecast a future values of system parameters (for example: network traffic, CPU utilization, number of network connections). There are a lot of publications about usage different types of ANN for network traffic prediction (See e.g.: [22], [23], [24], [25]) or intruder detection (See e.g.: [19], [20], [21]).

In our current research we choose the simplest artificial neural network – Multilayer Perceptron (MLP) for prediction of traffic time series values.

An MLP is a network of neuron called perceptrons. The perceptron is a binary classifier which compute a single output from multiple inputs (and the 'bias', a constant term that does not depend on any input value) as function of its weighted sum.

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + w_0 b\right) \quad (14)$$

where:

y is the output value

φ is activation function

w is weight vector

x is input vector

b is bias

MLP is a feedforward artificial neural network model consisting of layers of perceptrons, that maps sets of input data onto a set of appropriate output (see: Figure 11).

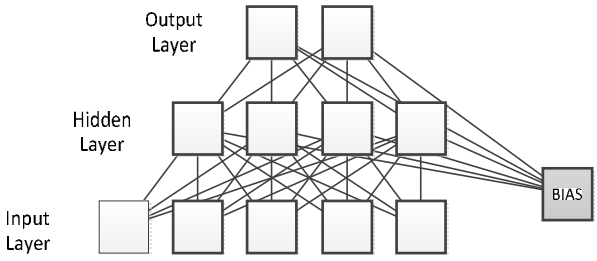

Figure 11: Overall look of MLP. Source: own research.

There are a few possible MLP architectures which can be used for time series prediction. One can use output layer with single neuron and its output value can be interpreted as the predicted value of the time series in the next moment or output layer with a group of neurons, which represent predicted time series values in a few next moments. The input layer can consist of different number of neurons too. When modelled time series has periodical character it seems to be good idea to set the number of input neuron equal to the period length or as multiple of the length, but when the whole period consists of big number of observations (i.e. our series day has 144 and week – 1008 observations), the ANN constructed in this way may be too big. Number of hidden layers and number of neurons in the hidden layers may be arbitrarily preselected or automatically set by ANN emulator.

In the research we decided to use the architectures with input neurons concerning the investigated time series value delayed by one, two, three measurements, one day, one day and one measurement, one week and one week and one measurement, one hidden layer and one output neuron. The network architecture were automatically optimized by adding input neurons (the neurons that did affect to output value).

12 Results and Conclusions

We decided to collect network traffic data from a few small- and middle-sized networks, described in the Table 1.

W1	Amateur campus network consisting of circa 25 workstrations. IDS has worked on the router which act also as the gateway to the Intenret as well as a few servers (www, ftp etc.).
T2	Campus network provided by middle-size IAP (about 400 clients)
T3	A network in a block of flats; one of networks mentioned in T1, containing about 20 clients
MM	Home network connected to the campus amateur network (with maximum speed of inbound traffic set on the bandwidth manager to 4 Mbps. The home network consist of five computers and two servers protected by firewall.
II	Local Area Network in small company (about 40 computers, two intranet servers).

Table 1: Investigated networks description (detailed information about these networks and descriptive statistics of collected time series are described in [9]). Source: own research.

Series	Protocol	Holt-Winters	MLP Topology	MAE/M
W1	TCP	45,92	2-1-1 (-1,-3)	46,18
W1	UDP	30,19	1-2-1 (-1)	31,73
W1	ICMP	31,27	4-2-1 (-1, -2, -3, -144)	34,54
T2	TCP	4,19	1-1-1 (-1)	4,23
T2	UDP	15,87	5-1-1 (-1, -2, -3, -144, -1009)	15,41
T2	ICMP	8,53	2-2-1 (-1, -2)	8,66
T3	TCP	4,11	1-1-1 (-1)	4,07
T3	UDP	15,45	1-1-1 (-1)	15,05
T3	ICMP	8,69	3-1-1 (-1, -3, -1008)	8,91
MM	TCP	64,40	2-2-1 (-1, -2)	75,72
MM	UDP	28,88	4-1-1 (-1, -3, -145, -1009)	30,12
MM	ICMP	10,57	3-1-1 (-1,-2,-3)	10,93
II	TCP	36,42	2-1-1 (-1,-2)	41,14
II	UDP	49,55	5-1-1 (-1, -2, -3, -144, 1008)	48,42
II	ICMP	110,65	1-1-1 (-1)	116,44

Table 2: Models fit. Source: own research.

Detailed results (in percent) of Holt-Winters models from the previous research and the MLP from the current one are shown on the Table 2. The "topology" column describes structure of particular MLP: number of neurons in input, hidden and output layer (f.e. "3-2-1" means "three input, two hidden and one output neuron") and information about delayed variables in input layer (f.e. "-1, -145" means: delayed by one measurement on the first input and delayed by 145 measurements on the second one; because we use time series with 10-minutes interval 144 means one day and 1008 means one week).

As one can see ANN appears to be promising solutions for traffic modelling. In the most of cases its fit is similar to the Holt-Winters Model and to the other models form our previous research. In the future works we plan to develop appropriate anomaly detection algorithm for MLP model and implement it as an additional model in profile generator. Also we plan to test another ANN models and architectures to improve the fit of our models.

13 Direction for future research

At the moment the most needed improvement to out program is to use a database for logging network traffic parameters instead of flat comma separated values file. For short logging time interval log file would grow rapidly and in the course of time access to log data will raise. Usage of database would have one other more major advantage – obtaining a needed sub-collection of log data will be easier and faster. Moreover by not using file for log data there should be lower memory and disk usage consumption – actually all data from log file are loaded into memory during forecasts calculations. With simple SQL queries there would be no need to do this – only data for current counter (time series) are necessary.

Second awaited development is use of NetFlow / IPFIX standard in storing and calculating network data. By this it would be simple to collect network data from many observation points. Afterwards device which support IPFIX protocol can filter and aggregate data and send it to Anomaly Detection server for further analysis. Implementation of IPFIX protocol would be good starting point for further improvements such as flow or route analysis (see e.g. [26] [27]).

References

[1] J. D. Brutlag, "*Aberrant Behavior Detection in Time Series for Network Monitoring*" 14th System Administration Conference Proceedings, New Orleans 2000, pp. 139-146

[2] J. Koziol, "*Intrusion Detection with Snort"*, Sams Publishing, Indianapolis, 2003

[3] R. Rehman, "*Intruder Detection with Snort"*, New Jersey 2003

[4] M. Skowroński, R. Wężyk, M. Szmit, "*Preprocesory detekcji anomalii dla programu Snort"* [inw:] Sieci komputerowe. T. 2. Aplikacje i zastosowania, Wydawnictwa Komunikacji i Łączności, Gliwice 2007, pp. 333-338

[5] M. Szmit, R. Wężyk, M. Skowroński, A. Szmit, "*Traffic Anomaly Detection with Snort"* [in:] Information Systems Architecture and Technology. Information Systems and Computer Communication Networks, Wydawnictwo Politechniki Wrocławskiej, Wrocław 2007, pp. 181-187

[6] M. Skowroński, R. Wężyk, M. Szmit, "*Detekcja anomalii ruchu sieciowego w programie Snort,"* „Hakin9" Nr 3/2007, pp. 64-68

[7] M. Szmit, A. Szmit, Usage of Modified Holt-Winters Method in the Anomaly Detection of Network Traffic: Case Studies, Journal of Computer Networks and Communications, vol. 2012, DOI:10.1155/2012

[8] M. Szmit, A. Szmit, "*Use of Holt-Winters method in the analysis of network traffic. Case study*", Springer Communications in Computer and Information Science vol. 160, pp. 224-231.

[9] M. Szmit, "Využití nula-jedničkových modelů pro behaviorální analýzu síťového provozu", [in:] Internet, competitiveness and organizational security, TBU, Zlín 2011

[10] The R Project for Statistical Computing Homepage http://www.r-project.org/

[11] P. Biecek, "*Przewodnik po pakiecie R"*, Gewert i Skoczylas, 2011, Partly available on www http://www.biecek.pl/R/Rwydanie2.pdf

[12] Ł. Komsta, "*Wprowadzenie do środowiska R ",* 2004, Available: http://cran.r-project.org/doc/contrib/Komsta-Wprowadzenie.pdf

[13] P. Teetor, "*R Cookbook"*, O'Reilly Media, 2011

[14] P. Teetor, "*25 Recipes for Getting Started with R"*, O'Reilly Media, 2011

[15] Szmit Maciej, Adamus Sławomir, Bugała Sebastian, Szmit Anna: Anomaly Detection 3.0 for Snort(R), [in:] SECURITATEA INFORMAȚIONALĂ 2012, pp. 37-41, Laboratorul de Securitate Informațională al ASEM, Chişinău 2012

[16] M. Szmit, A. Szmit, Usage of Pseudo-estimator LAD and SARIMA Models for Network Traffic Prediction. Case Studies, Communications in Computer and Information Science, 2012, Volume 291, 229-236, DOI: 10.1007/978-3-642-31217-5-25

[17] M. Szmit, Modelování, simulace a behaviorální analýza procesů síťového provozu jako výzkumné metody plánování efektivního využití síťového provozu, [in:] Internet, competitiveness and organizational security, pp. 139-144, Tomas Bata University, Zlín 2012

[18] S. Gelper, R. Fried, C. Croux, "Robust forecasting with exponential and Holt–Winters smoothing" [in]: Journal of Forecasting, Volume 29, Issue 3, pp. 285–300, April 2010

[19] B. Guzik, D. Appenzeller, W. Jurek, Prognozowanie i symulacje. Wybrane zagadnienia, Wydawnictwo AE w Poznaniu, Poznań 2004

[20] P. Goodwin, “The Holt-Winters Approach to Exponential Smoothing: 50 Years Old and Going Strong”, FORESIGHT Fall 2010 pp. 30-34

[21] E.S. Gardner, Jr., Exponential Smoothing: The state of the art – Part II, International Journal of Forecasting, 22/2006, pp. 637-666.

[22] R. J. Hyndman, A. B. Koehler, J.K. Ord, R. D. Snyder, Forecasting with Exponential Smoothing: The State Space Approach, Springer, Berlin 2008

[23] P. Cortez, M. Rio, M. Rocha, P. Sousa: Multi-scale Internet traffic forecasting using neural networks and time series methods, Expert Systems: The Journal of Knowledge Engineering, (accepted paper, in press),http://onlinelibrary.wiley.com/doi/10.1111/j.14680394.2010.00568.x/abstract

[24] AnomalyDetection Homepage http://www.anomalygetection.info

[25] M. Skowroński, R. Wężyk, "Systemy detekcji intruzów i aktywnej odpowiedzi", Master Thesis, Politechnika Łódzka, 2004

[26] Byungjoon Lee, Hyeongu Son, Seunghyun Yoon, Youngseok Lee, “End-to-End Flow Monitoring with IPFIX” [in:] Lecture Notes in Computer Science, 2007, Volume 4773/2007, pp. 225-234, Available at: http://www.springerlink.com/content/l868g0x635324129/

[27] Youngseok Lee, Seongho Shin, Taeck-geun Kwon, “Signature-Aware Traffic Monitoring with IPFIX” [in:] Lecture Notes in Computer Science, 2006, Volume 4238/2006, pp. 82-91. Available at: http://www.springerlink.com/content/w312715821374007/

[28] James W. Hong, Sung-Uk Park, Young-Min Kang, Jong-Tae Park, “Enterprise Network Traffic Monitoring, Analysis, and Reporting Using Web Technology” [in:] Journal of Network and Systems Management Volume 9, Number 1 (2001), pp. 89-111.

[29] Miroslaw Malek, Bratislav Milic, Nikola Milanovic, "Analytical Availability Assessment of IT Services" [in:] Lecture Notes in Computer Science, 2008, Volume 5017/2008, pp. 207-224.

[30] A. N. Nazarov, M. M. Klimanov, "Estimating the informational security level of a typical corporate network".

[31] J. Gómez, C. Gil, N. Padilla, R. Baños, C. Jiménez, "Design of a Snort-Based Hybrid Intrusion Detection System" [in:] Lecture Notes in Computer Science, 2009, Volume 5518/2009, pp. 515-522.

[32] Joshua Ojo Nehinbe, "A Simple Method for Improving Intrusion Detections in Corporate Networks" [in:] Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2010, Volume 41, pp. 111-122.

[33] Nathalie Dagorn, "Cooperative Intrusion Detection for Web Applications" [in:] Lecture Notes in Computer Science, 2006, Volume 4301/2006, pp. 286-302.

[34] Kulesh Shanmugasundaram, Nasir Memon, "Network Monitoring for Security and Forensics" [in:] Lecture Notes in Computer Science, 2006, Volume 4332/2006, pp. 56-70.

[35] S. Adamus, S Bugała, “Some aspects of network anomaly detection”, Master Thesis (in Polish), Technical University of Lodz, 2012

[36] M. Szmit, S. Adamus, S. Bugała, A. Szmit: “Implementation of Brutlag's algorithm in Anomaly Detection 3.0”, Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 685–691, PTI, IEEE, Wrocław 2012, IEEE Catalog Number CFP1285N-USB, ISBN:978-83-60810-51-4