Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 685–691

ISBN 978-83-60810-51-4

Implementation of Brutlag's algorithm in Anomaly Detection 3.0

Maciej Szmit
Technical University of Łódź, Computer Engineering Department, 18/22 Stefanowskiego Street, 90-924 Łódź, Poland
Orange Labs Poland, 7 Obrzeżna Street, 02-691 Warsaw, Poland
Email: maciej.szmit@gmail.com

Anna Szmit
Technical University of Łódź, Department of Management, 266 Piotrkowska Street, 90-924 Łódź, Poland
Email: agorecka@p.lodz.pl

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

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

***Abstract*—This paper presents information about Anomaly-Detection – a Snort-based network traffic monitoring tool. The article concerns use of based on Holt-Winters forecasting method in real-time behavioral analysis of network traffic.**

I. 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 demonstrates the capabilities of the tool 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 Anomaly Detection 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. Fig 1 shows data flow diagram for AD.

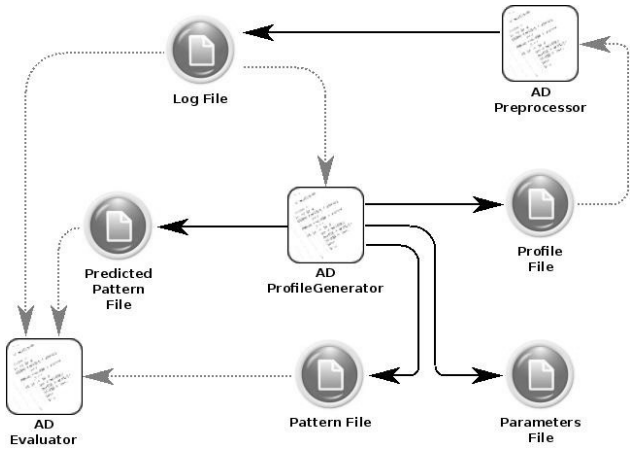

Fig 1. Anomaly Detection data flow diagram. Source: [15]

Black solid arrows means saving to file and gray 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.

II. 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 Anomaly Detection 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 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' 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 lane read into the structure at a time, this gives possibility to dynamically alter the profile file, whose forecast length (amount of lines) can by any. 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.

III. Profile Generator

The current version of Profile Generator (see e.g. [7] [8] [9]) is based on R language / environment (The R Project for Statistical Computing) (see e.g. [10] [11] [12] [13] [14]). 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 fur-

ther calculations and other parts – depending on the given parameters – calculates future network traffic forecasts.

To use all functions of Profile Generator local R-project installation must contain listed above additional packages. They can be installed in R with commands:

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:

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

Naive method:

$$\hat{y}_t = y_{t-T} \tag{2}$$

where T is day or week period

or

$$\hat{y}_t = y_{t-1} \tag{3}$$

Autoregressive time series model:

$$\hat{y}_t = a_0 + a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_k y_{t-k} \tag{4}$$

Holt-Winters model:

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

where:

L is level component given by:

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

P is trend component given by:

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

S is seasonal component given by:

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

Brutlag method:

$$\hat{y}_{max_t} = L_{t-1} + P_{t-1} + S_{t-T} + m \cdot d_{t-T} \tag{9}$$

$$\hat{y}_{min_t} = L_{t-1} + P_{t-1} + S_{t-T} - m \cdot d_{t-T} \tag{10}$$

where:

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

d is predicted deviation given by:

$$d_t = \gamma|y_t - \hat{y}_t| + (1-\gamma)d_{t-T} \tag{11}$$

where:

k is number of measurements in time series

t is moment in time

$\hat{y}_t$ 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

IV. 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=profile.config.frequency, start=c(as.numeric(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 and/or β and/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.

For testing purposes total number of TCP packets from one exemplary network was used. Testing data were gathered for few weeks (between January 1st and March 11th). Illustration Fig 2 shows one weekly period (from January 1st to January 7th) of testing data.

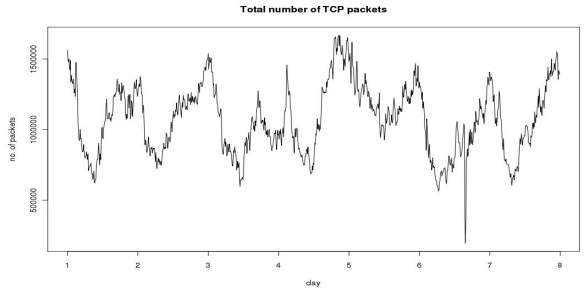

Fig 2. One period of testing data.

`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. Fig 3 shows those decomposed data.

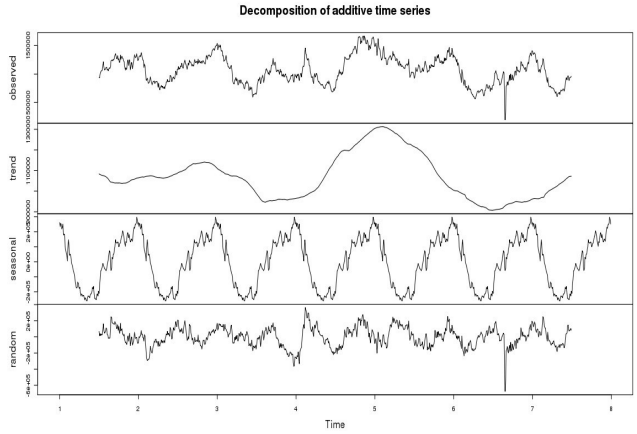

Fig 3. Decomposed time series.

`HoltWinters()` function estimates HW model smoothing parameters (alpha, beta and gamma), which were for testing data as follows:

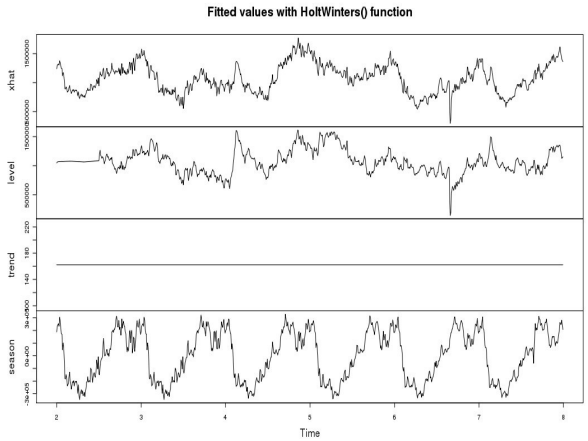

Fig 4. Fitted Holt-Winters.

alpha: 0.8140128
beta : 0
gamma: 1

Fig 4 shows fitted time-series of Holt-Winters model.

Fitted values compared to observed values for given testing data:

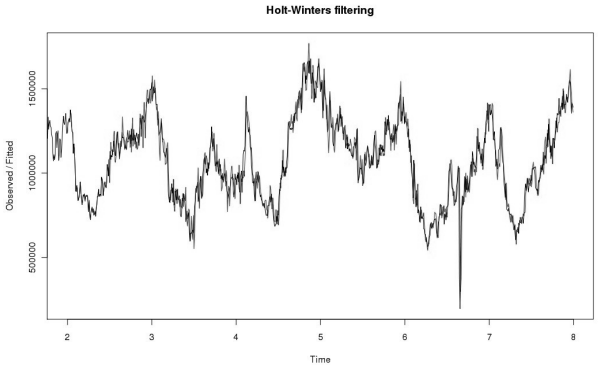

Fig 5. Holt-Winters fitted to observed comparison.

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 illustration Fig 6.

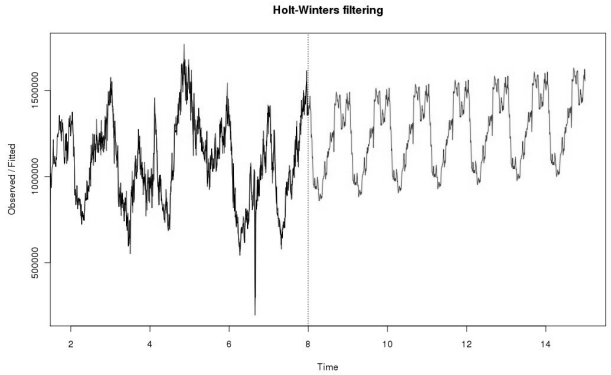

Fig 6. Holt-Winters prediction.

V. Brutlag method

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.

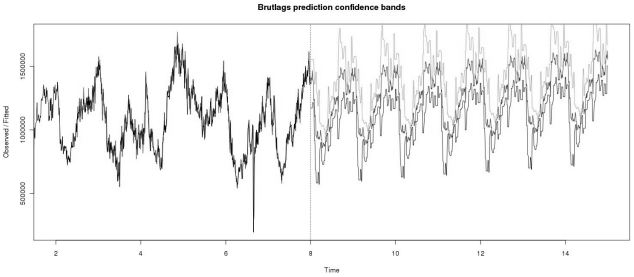

Fig 7. Brutlags confidence bands.

Illustration Fig 7 shows computed confidence bands for HW time series prediction. 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

VI. Usage of Profile Generator

Profile 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, 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
```

VII. Profile 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 statis-

tic $\frac{MAE}{M}$ 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.

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

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

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

VIII. Usage of Profile Evaluator

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
```

IX. Conclusion

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 Processdings, New Orleans 2000, pp. 139-146, Available: http://www.usenix.org/events/lisa00/full_papers/brutlag/brutlag_html/

[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."* (Article in press [in:] Journal of Computer Networks and Communication) Available at: http://www.hindawi.com/journals/jcnc/aip/192913/

[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. Available at: http://www.springerlink.com/content/w332871210136544/

[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: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] M. Szmit, S. Adamus, S. Bugała, A. Szmit, "AnomalyDetection 3.0 for Snort" [in:] Conference Proceedings of SECURITATEA INFORMAŢIONALĂ 2012, Chişinău, Moldova. To be published.

[16] M. Szmit, A. Szmit, "Usage of Pseudo-estimator LAD and SARIMA Models for Network Traffic Prediction. Case Studies." (accepted for publication in Springer Communications in Computer and Information Science; Conference Computer Networks, 2012)

[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,Available:http://www.forecasters.org/pdfs/foresight/free/Issue19_goodwin.pdf

[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. Available at: http://www.springerlink.com/content/k4917220vxpw1765/
[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. Available at: http://www.springerlink.com/content/u415087711752925/
[30] A. N. Nazarov, M. M. Klimanov, "Estimating the informational security level of a typical corporate network". Available at: http://www.springerlink.com/content/2646q516161xp66v/
[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. Available at: http://www.springerlink.com/content/83h7164gn7q240j6/
[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. Available at: http://www.springerlink.com/content/h644301061760174/
[33] Nathalie Dagorn, "Cooperative Intrusion Detection for Web Applications" [in:] Lecture Notes in Computer Science, 2006, Volume 4301/2006, pp. 286-302. Available at: http://www.springerlink.com/content/b4mt420gk1077617/
[34] Kulesh Shanmugasundaram, Nasir Memon, "Network Monitoring for Security and Forensics" [in:] Lecture Notes in Computer Science, 2006, Volume 4332/2006, pp. 56-70. Available at: http://www.springerlink.com/content/jv1250p116480708/