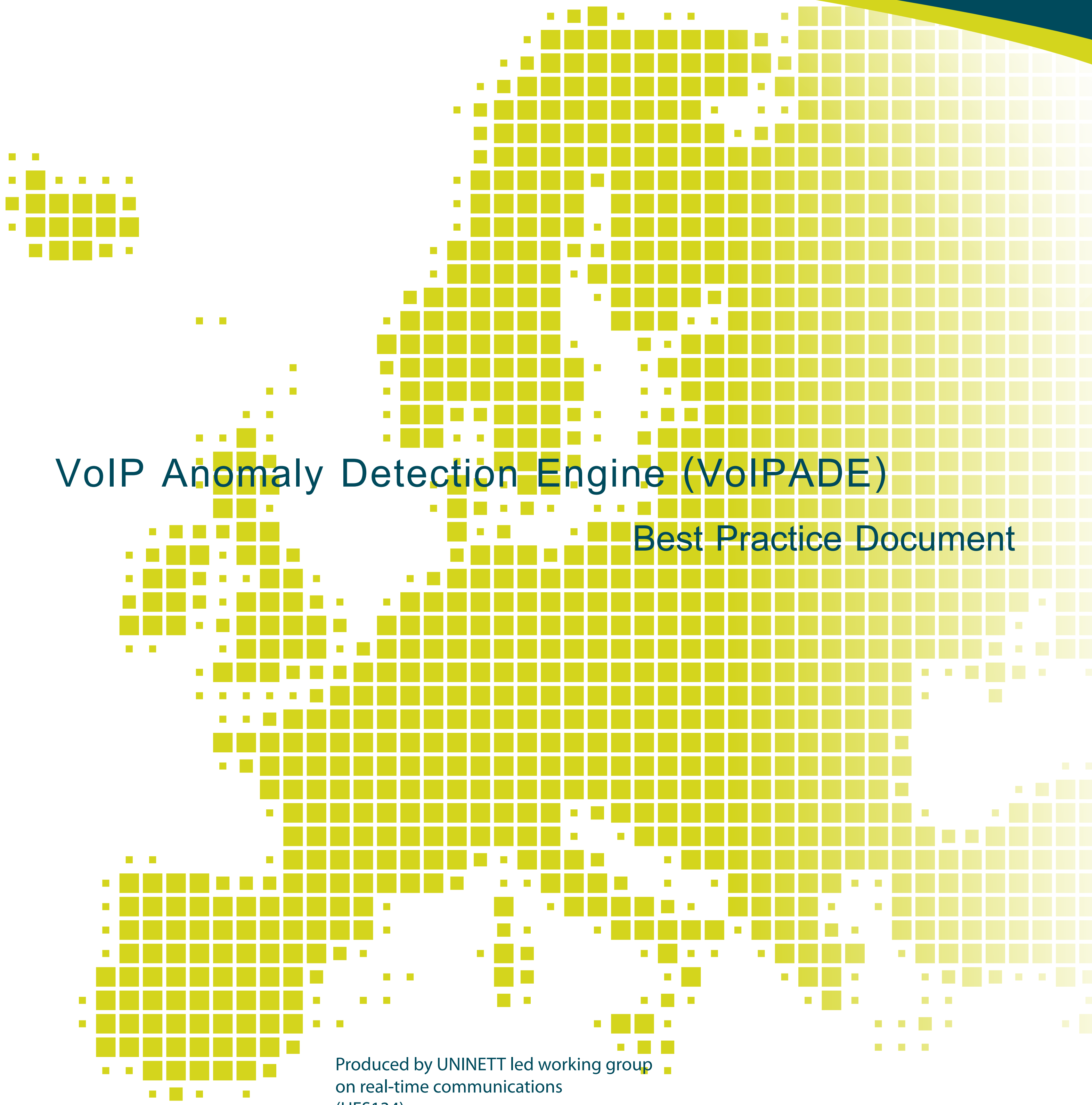
VoIP Anomaly Detection Engine (VoIPADE)

Best Practice Document

Produced by UNINETT led working group
on real-time communications
(UFS134)

Author: Gurvinder Singh
August 2011



Document No:	GN3-NA3-T4-UFS134
Version / date:	August 2011
Original language :	Norwegian
Original title:	"VoIP Anomaly Detection Engine"
Original version / date:	2011-08-02
Contact:	campus@uninett.no

UNINETT bears responsibility for the content of this document. The work has been carried out by an UNINETT led working group on real-time communications as part of a joint-venture project within the HE sector in Norway.

The research leading to the results of this report has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 238875, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3)'.

Table of Contents

1 Introduction

In recent years the Voice over Internet Protocol (VoIP) has gained a lot of attention, due to its benefit to make cheaper voice calls compared to the traditional PSTN network. VoIP mainly uses the Session Initiation Protocol (SIP) with RTP to provide the required functionality. It makes various services e.g. Voice conferencing, IVR etc easily accessible compared to the PSTN network.

Similar to any other system on the connected world, VoIP systems are also susceptible to the attacks by remote/local attacker. The attacker can abuse the system by launching an attack on the system, generally to make expensive calls, which results in a huge sum of bill in a short span of the attacked period. Such abnormal calls can have either high frequency to the expensive destination e.g. International or Toll numbers, or a few number of calls for longer duration where user has to pay large amount of money per second.

VoIP Anomaly Detection Engine (VoIPADE) is implemented by UNINETT and is a detection engine, which detects anomaly in the outgoing calls pattern. Thus VoIPADE detects such abnormal calls which can have either high frequency to expensive destination or few numbers of calls, which lasts for longer duration to the expensive numbers.

To detect anomaly in the call pattern VoIPADE uses a behavioural learning algorithm. The algorithm learns the behaviour of organization's call pattern. It trains the algorithm for a specified training period and drives a threshold value for the organization's call traffic. After the training phase VoIPADE enters the detection phase, where it uses the derived threshold value to detect the anomalies. It keeps learning the call pattern even in the detection phase, but only from the behaviour of the normal call pattern. The learning in the detection phase helps the threshold value to reflect the legitimate changes in the call pattern and reduce false positives.

UNINETT is currently deploying a national SIP infrastructure for its customers. The infrastructure contains Session Border Controllers (SBCs) that provide a common SIP-based gateway to PSTN. UNINETT has found it important to implement a VoIP anomaly detection system in order to efficient detect and stop unwanted calls. VoIPADE has been in pilot at UNINETT since April 2011 and has successfully given alerts on several occasions.

VoIPADE can be installed on any VoIP system that logs data using CDRs (Call Detail Records).

2 The VoIPADE Architecture

VoIPADE consists of 5 different modules to perform its functionality. The different modules are shown below in the Figure 1.

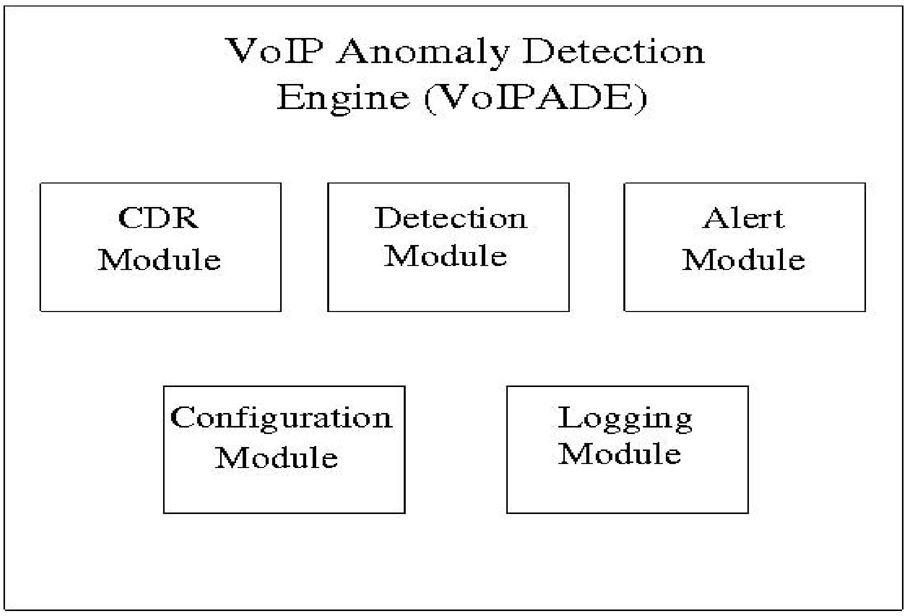


Figure 1: VoIPADE Architecure

We elaborate on the modules in the following chapters.

2.1 The CDR Module

The CDR module is responsible for making the connection to the CDR database and fetches data from it by making a given query. It fetches the CDR records related to all the monitored call types, which are placed in time interval from the last fetched records. The CDR records are logged by Asterisk or any other similar kind of platform. It contains the detail record of the placed call. VoIPADE uses these call records to detect the anomaly in the call pattern of the organization.

The CDR module needs the following fields with the exact names and type to exist in the CDR database schema to operate correctly. The CDR database can have other fields too, but the fields mentioned below are mandatory. The order of fields mentioned below can be random:

- id serial,
- calldate timestamp with time zone,

- src text NOT NULL,
- dst text NOT NULL,
- billsec integer NOT NULL,
- accountcode text NOT NULL,
- calltype text not null

id represents the unique id for a CDR record, *calldate* tells us the time and date when the given call was placed, *src* tells us who is the caller. *dst* tells us the destination to which the call is placed, *billsec* tells us the duration of the call which will be billed by the telephony service provider, *accountcode* tells us the institution from where the call has been originated and lastly *calltype* tells us what is the type of destination number.

Different types of calls are:

- INTERNATIONAL – calls which are made to numbers out of the country.
- MOBILE – calls which are made to the mobile number inside the country.
- PREMIUM – calls which are made to the toll services number which costs large amount of money. They are the calls to the national toll services numbers.
- SERVICE – calls which are made to the toll services number which does not cost money. They are the calls to the national toll services numbers.
- DOMESTIC – calls which are made to the fixed line national numbers.
- EMERGENCY – calls made to the emergency services such as fire brigade, police or ambulance.

Note that the call type names mentioned above are case sensitive, so while classify and marking the calls to their corresponding types, check the call type name to be of correct case.

2.2 The Detection Module

After fetching the data from the CDR database, the CDR data is sent to the detection module. This module uses the behavioural based algorithm to detect anomaly in the fetched data. The algorithm used in the VoIPADE to detect the anomalous behaviour is "*Hellinger Distance*" algorithm[1]. In this algorithm we calculate the distance between the chosen parameters, which are the different call types. The distance value is a measure of the variability between the two probability distributions of the parameters. The value is calculated between the *probabilities* of occurrences of desired event during *training period* and during the *testing period*.

$$HD = (\mathrm{sqrt}(p) - \mathrm{sqrt}(q))^2$$

where *p => probability of the desired event during the training period*
q => probability of the desired event during the testing period

p,q is calculated as *number of desired event occurrences* divided by *total number of events* in the given time period. The desired events are the call frequency and call duration of *different call types*. The section 4.1 in [1] gives good overview of using "Hellinger Distance" algorithm for anomaly detection. The distance value is calculated for both frequencies of the calls and their duration in the given interval. This helps in detecting the attack related to high frequency calls to expensive destination e.g. International calls, whereas the duration values helps in detecting the attack which are related to few calls with large duration to expensive destinations, e.g. Toll fraud numbers.

From the calculated distance value, we use the stochastic gradient algorithm to compute the dynamic threshold value. The threshold value is based upon the calculated distance value and derived using the the Jacobson's Fast algorithm for RTT mean and variation calculation [2].

$$Err = m_n - a_{n-1}$$
$$a_n = a_{n-1} + g * Err$$
$$v_n = v_{n-1} + h.\ (|Err| - v_{n-1})$$

Err is the difference between measured HD value (m_n) from this interval and expected HD value (a_n), v_{n-1} and v_n represent the previous and current mean deviations. *g* and *h* are the coefficient values to make the expected value of HD with the help of mean deviation to converge as close as possible to the actual HD value. The default values for g = $1/2^3$ and h = $1/2^4$. The threshold value is then calculated using the HD value (a_n) and mean deviation (v_n) as shown below:

$$\{Hd_{thres}\}^{n+1} = sensitivity * a_n + adaptability * v_n$$

sensitivity and *adaptability* value tells the algorithm, how much sensitive the threshold value should be to the changes and how much adaptable threshold should be to the call pattern changes. The value of *sensitivity* and *adaptability* should be specified in the configuration file according to the organization call traffic pattern.

The threshold value calculated after the training period is stored in the threshold database. The value is stored after each interval, so that in the case of system crash or engine crash, the threshold value can be stored back quickly. This avoids the unnecessary training on the start of the engine after crash. The threshold database schema to store the threshold value and other related fields is given in the Appendix A.

2.3 The Configuration Module

The configuration module is used to parse the parameters and their values defined in the configuration file. VoIPADE uses libyaml to parse the configuration file and get access to the defined value. The configuration file contains various parameters, which provide a great deal of flexibility in customizing the VoIPADE to the target specific needs of an organization. Please make sure while specifying values for the configuration parameters, the values are case sensitive. The following parameters are defined in the configuration file:

```
Institution name for which we are running the anomaly detection engine.
institution: 59713
```

The run mode options for the VoIPADE engine are either "online" or "offline". In online mode VoIPADE will run continuously, until it will be killed or system has been crashed. In the offline mode, it will read the given database for the CDR records and will detect the attacks, until the provided ending-date has been reached and after that VoIPADE will shut down itself.

```
run-mode: offline
ending-date: '2011-04-01 00:00:00'
```

The information to make a connection to the CDR. The engine will fetch the cdr records and run the anomaly detection algorithm on them.

```
cdr-database:
host: localhost
username: asterisk
password: 123456
database-name: asterisk
table: cdr
port: 3306
```

Similar to the CDR database, the connection information for the Alert and Threshold database should be specified in the configuration file.

The logging level for the VoIPADE detection engine. The options are “info”, “debug” or “error”. I.e:

```
logging-mode: info
```

The alert mode configuration instructs VoIPADE engine where to send alerts. The options are “syslog”, “hobbit” or “both”. In case of *hobbit/both* define the alert file full path and name as provided in the hobbit server configuration.

```
alert-mode: hobbit
alert-file: /var/log/voip_alert.log
```

The *call-type* parameter specifies for what types of calls you would like to analyze in the detection engine. The options are "International", “Mobile”, “Premium”, “Service”, “Domestic” and ”Emergency". If you want to run the VoIPADE engine for all call types then specify call-type as "All".

```
call-type: "International,Mobile,Premium"
```

The *ad-algo* parameter contains various parameters which are specific to the anomaly detection algorithm. The *sensitivity* value tells the engine how much sensitive the engine should be to the changes in the call pattern. Its value should be higher than 1.0. As a general rule, if you want the engine to be highly sensitive to call pattern changes, then sensitivity should be close 1.0. The higher the value of sensitivity is, the lower the sensitivity is to the call pattern changes. If there is considerable percentage of false positive in the alerts raised by VoIPADE, then the solution would be to increase the sensitivity value from its previous value.

The *adaptability* value tells the engine how to adapt to small changes in the call traffic behaviour of the monitoring institution. Its value should be in the range of [0.0-1.0]. The adaptability value close to 0.0 means, the system is not much adaptive to the changes in the traffic and where the value near to 1.0 means system is highly adaptive. As a general rule, the adaptability value should be somewhere in the range of 0.2-0.8 depending upon the call traffic pattern.

The *interval* value (in minutes) is the time span after which the engine scans the placed calls in the duration from last scanned time plus the specified interval. The VoIPADE engine will fetch the CDR records during the specified time span and will run the anomaly detection algorithm on the fetched records.

The *threshold-restore* field tells the engine if it should try to restore the threshold value from the threshold database or not. This will help VoIPADE in a quick restart after a crash or system reboot. VoIPADE stores the threshold value after it completes the training for the first time and keeps storing the updated threshold value after each interval. If this option is set to yes, then upon restart the engine will fetch the last threshold value and will enter in to detection phase directly without any need of training the detection algorithm.

The *call-freq* field tells the engine about minimum number of calls which are allowed for the organization in a time span as specified in the *interval* field. Similarly *call-duration* field tells engine about the minimum duration (in minutes) of the calls allowed for the organization in the time span equal to *interval*. These two fields are application to the total number of all the call types specified in the *call-type* field. If the call duration and call frequency of all the monitored call types are less than the specified values, then the engine will not run the detection algorithm and thus avoids the unnecessary overhead from the engine.

```
ad-algo:
sensitivity: 1.3
 adaptability: 0.25
 interval: 10
 threshold-restore: 'no'
 call-freq: 10
 call-duration: 10
```

The *call-duration* parameters specify values in minutes of each paid call. These values signify that the total duration of the corresponding call type is allowed, if the duration value is less than the specified value in the configuration value. These values are used to avoid unnecessary false positives during office working hours.

```
call-duration:
 mobile: 3600
 premium: 3600
 international: 3600
```

The *office-time* field tells what the office working hours are. This helps the engine to be more lenient to the traffic spikes during working hours. This helps reducing the false positive rate. The working hours value for starting and ending time hours should be specified in 1-24 hour clock format.

```
office-time:
start_time: 8
end_time: 16
```

The *initial-timestamp* field tells the engine from what time the engine will start the training of the detection algorithm. If this field is not specified the engine will fetch the first CDR record from the CDR database and will start the training from the time stamp of the first CDR record.

The *training-period* field tells the engine for how long it should train the detection algorithm. The value should be specified in the configuration file in minutes and default value is 10800 (1 week). The *detection-start-ts* field tells the engine as from which time stamp, it should start the detection phase. This option is useful when there is a time gap between the training phase and the detection phase time stamps. If the detection phase has to be started right after the training phase, then this option can be commented.

```
initial-timestamp: '2010-01-01 02:03:36'
training-period: 10800
detection-start-ts: '2010-01-11 00:00:00'
```

2.4 The Alert Module

VoIPADE supports two methods to send alerts after anomalous behaviour has been detected. It can send alerts to *syslog* or to the *hobbit* [3] server or to *both*. The alert message to be sent to syslog or to the hobbit server contains *time stamp, status message, account code* and the *unique alert ID*. The example alert message will look like this

```
[2010-01-17 16:37:56]    FATAL  59713  1
```

if there has not been any alert in the given interval then only in the *hobbit* alert mode, it will log the OK status as

```
[2010-01-17 16:37:56]    OK  59713
```

This helps the Hobbit server to keep the service status updated. The information in the status message uniquely identifies each alert and the institution for which the alert has been raised. Based on the status message, an alert can be sent to the responsible person/administrator at the corresponding institution identified by the account code in the alert message. In the Hobbit server, it is easy to configure the method in which alerts will be delivered to the responsible person at the institution e.g. via SMS or email etc.

VoIPADE logs all the CDR records from which anomaly has been detected in the given interval. The records are logged in to the CDR alert database, the information to be stored in the alert database contains following fields.

- ser_id serial,
- alert_id integer DEFAULT 0 NOT NULL,
- cdr_id integer DEFAULT 0 NOT NULL,
- calldate datetime DEFAULT '0000-00-00 00:00' NOT NULL,
- src varchar(80) DEFAULT ' ' NOT NULL,
- dst varcar(80) DEFAULT ' ' NOT NULL,
- billsec integer DEFAULT 0 NOT NULL,
- calltype varchar(32) DEFAULT ' ' NOT NULL,
- accountcode varchar(20) DEFAULT ' ' NOT NULL,
- PRIMARY KEY(ser_id)

The field names are self explanatory and help the administrator to figure out the responsible number/numbers for the alert and take the action accordingly.

2.5 The Logging Module

The logging module logs the informative or error messages to stdout or stderr correspondingly. In the configuration file, the logging mode can specified and depending upon the mode, message belonging to specified mode will be output, except the error message which will always be output to the stderr, irrespective of the specified logging mode. The different logging modes supported by VoIPADE are:

1 info
2 debug
3 error

The names of the logging modes are self explanatory in themselves. The log messages will contain the time stamp when the message has been output. The info messages will look like this

```
[21/5/2010 -- 11:01:28] <INFO> Training the engine for detection of anomalous behaviour...
```

The debug messages will look like this

```
[21/5/2010 -- 11:01:38] <DEBUG> [util-detection.c:260] Duration of Total Calls: 0
```

The debug message contains the filename, line number from which the message has been printed. Lastly the error messages will look like this

```
[21/5/2010 -- 11:01:43] <ERROR> [util-detection.c:594] Failed in connecting to threshold-database
```

The error messages will also contain the filename and the line number, where the error has been occurred.

3 Traffic Analysis

Figure 2 below shows how the threshold value adapts itself to changes in the traffic. It learns from the distance value which reflects the changes in traffic. This allows the engine to reflect the traffic pattern changes and it reduces the number of false positives.

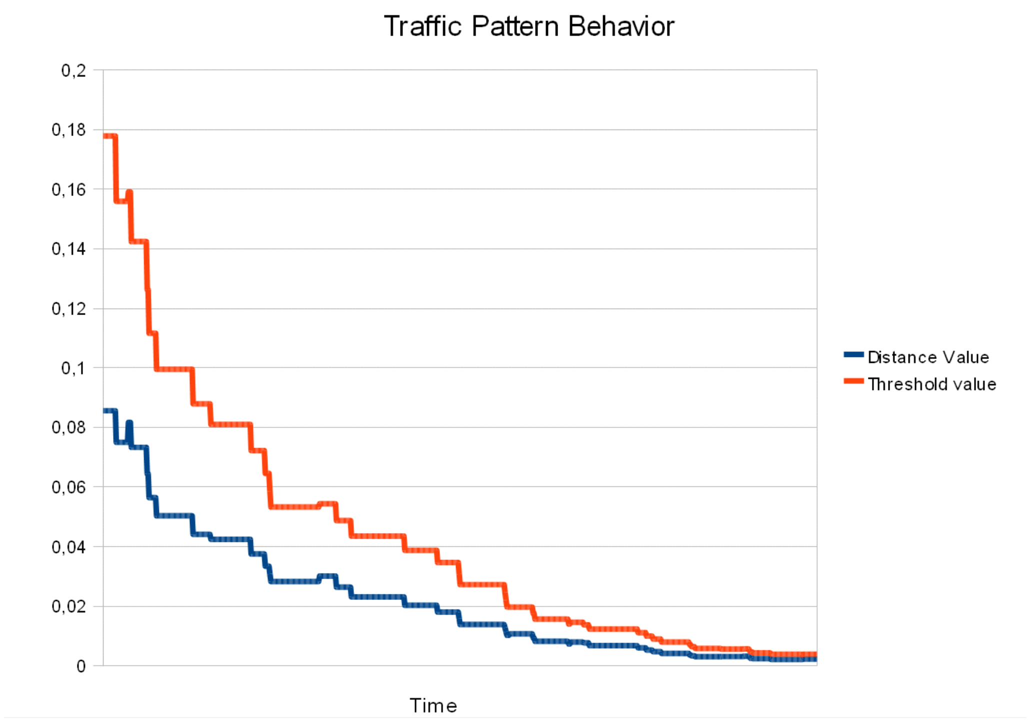


Figure 2: Threshold and distance value behaviour over time

Figure 3 shows the result from a case with real traffic at a Norwegian university. VoIPADE adapts to the spikes and learns from it. In the figure you will notice that the engine only take into consideration the traffic spikes that are within the threshold range. If the spike is result of an abnormal behaviour VoIPADE will not learn from it, which in turn makes it more resilient towards being tricked by an attacker.

Figure 4 shows the result from VoIPADE while examining the data from a real attack. The spikes in the starting part of the graph are the traffic spikes from attack. VoIPADE was able to detect the attack within 20 minutes after the attack was launched. The time interval can be reduced even further, if faster detection is required.

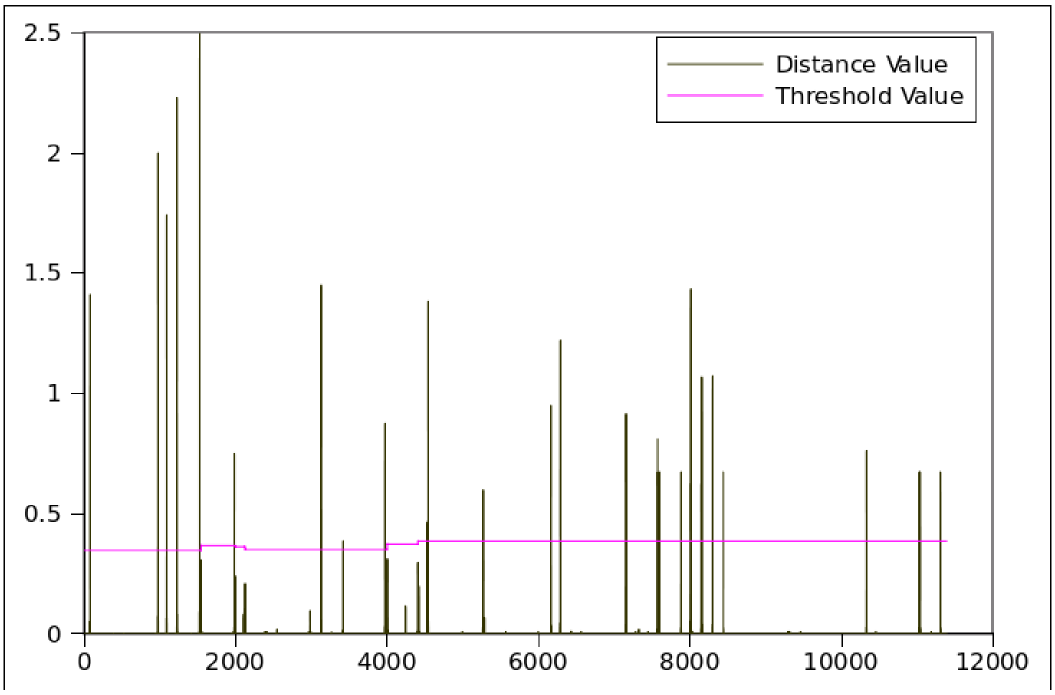


Figure 3: Adaptive Distance value behaviour for real traffic at a Norwegian university seen over time

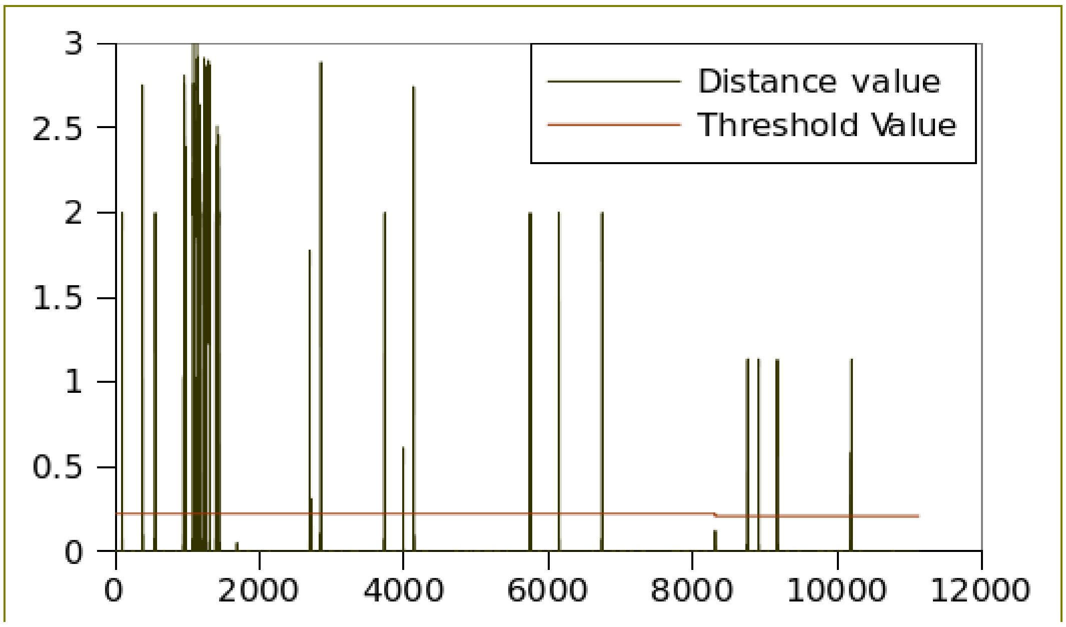


Figure 4: Distance value behaviour for real traffic where there was an attack

4 License

VoIPADE is available at http://software.uninett.no

Appendix A

Installation Procedure

The VoIPADE engine requires **libmysqlclient-dev** and **libyaml-dev** libraries to build. The libmysqlclient-dev is used for the mysql database communication and libyaml-dev is used for parsing the configuration file.

To install these libraries on you Ubuntu system, run the command as

$ sudo apt-get -y install libyaml-dev libmysqlclient-dev

This will install the required libraries for the VoIPADE. Now switch the VoIPADE command directory and run the command as

$ make

If no error has been reported during the build process, then to install the VoIPADE into the default location, run the command as

$ sudo make install

This will install the VoIPADE binary in to the /usr/local/sbin as "**VoIPADE**" and store the default configuration file in to the /usr/local/etc/VoIPADE with filename as "**VoIPADE.yaml**". Now VoIPADE has been installed and ready to detect the anomalies, if any !!

To run the VoIPADE, it needs three postgresql databases: **CDR database, voipade_alert database and voipade_threshold database**. For the CDR database, the require fields are mentioned in the CDR module section. Make sure that you have all the required field in your CDR database. The VoIPADE will make the consecutive queries for all the records in the given interval, so to enhance the performance it is a good idea to create the index for calldate/start field too. To do so run the following command on the database command prompt:

create index ts_idx on cdr(calldate);

This will increase the performance of VoIPADE drastically, as VoIPADE will be able to fetch the data from the database by spending much less time in wait. This happens due to the creation of binary index instead of searching/fetching data serially.

For the voipade_alert database, the database schema is given in the alert module, create the table by running the following command on the mysql command prompt as:

```
CREATE TABLE voipade_alert (
 ser_id serial,
 alert_id integer DEFAULT 0 NOT NULL,
 cdr_id integer DEFAULT 0 NOT NULL,
 calldate datetime DEFAULT '0000-00-00 00:00:00' NOT NULL,
 src varchar(80) DEFAULT ' ' NOT NULL,
 dst varchar(80) DEFAULT ' ' NOT NULL,
 billsec integer DEFAULT 0 NOT NULL,
 calltype varchar(32) DEFAULT ' ' NOT NULL,
 accountcode varchar(20) DEFAULT ' ' NOT NULL,
 PRIMARY KEY(ser_id)
);
```

This will create the voipade_alert data base. The last data base is to store the threshold values from each interval to make the restoration of the engine possible. To create the threshold database, run the command as:

```
CREATE TABLE voipade_threshold (
 threshold_id serial,
 num_int integer DEFAULT 0 NOT NULL,
 dur_int integer DEFAULT 0 NOT NULL,
 p_fint double precision DEFAULT 0.0 NOT NULL,
 p_dint double precision DEFAULT 0.0 NOT NULL,
 num_mob integer DEFAULT 0 NOT NULL,
 dur_mob integer DEFAULT 0 NOT NULL,
 p_fmob double precision DEFAULT 0.0 NOT NULL,
  p_dmob double precision DEFAULT 0.0 NOT NULL,
 num_prem integer DEFAULT 0 NOT NULL,
 dur_prem integer DEFAULT 0 NOT NULL,
 p_fprem double precision DEFAULT 0.0 NOT NULL,
 p_dprem double precision DEFAULT 0.0 NOT NULL,
  num_ser integer DEFAULT 0 NOT NULL,
 dur_ser integer DEFAULT 0 NOT NULL,
 p_fser double precision DEFAULT 0.0 NOT NULL,
 p_dser double precision DEFAULT 0.0 NOT NULL,
  num_dom integer DEFAULT 0 NOT NULL,
 dur_dom integer DEFAULT 0 NOT NULL,
 p_fdom double precision DEFAULT 0.0 NOT NULL,
 p_ddom double precision DEFAULT 0.0 NOT NULL,
  num_emr integer DEFAULT 0 NOT NULL,
 dur_emr integer DEFAULT 0 NOT NULL,
 p_femr double precision DEFAULT 0.0 NOT NULL,
 p_demr double precision DEFAULT 0.0 NOT NULL,
 num_total bigint DEFAULT 0 NOT NULL,
 dur_total bigint DEFAULT 0 NOT NULL,
 dist_value double precision DEFAULT 0.0 NOT NULL,
 mean_dev double precision DEFAULT 0.0 NOT NULL,
 threshold double precision DEFAULT 0.0 NOT NULL,
 last_ts datetime DEFAULT '0000-00-00 00:00:00' NOT NULL,
 accountcode varchar(30) NOT NULL,

 PRIMARY KEY(threshold_id)
);
```

Now as all the required databases are in place, to run the VoIPADE engine type the command as

```
$ VoIPADE
```

This will start the engine and you will see the output as

```
[25/5/2010 -- 09:41:48] <INFO> Training the engine for detection of anomalous behavior...
[25/5/2010 -- 09:41:49] <INFO> VoIP Anomaly Detection Engine has been started successfully...
```

These messages show that the engine has been started successfully and been in place to detect the anomalies in the call pattern. You can also specify the different configuration file by passing the file path with *-c* parameter as.

$ VoIPADE -c <path to configuration file>

To generate the documentation for the VoIPADE code, switch to the *docs* directory in the voipade source folder and run the command as

$ doxygen VoIPADE_docs.conf

The above command needs doxygen program to be installed on your system to work. This will generate the *htm* files in the *doxygen* folder under the *docs* directory. Open the *index.htm* file, this will have the links to all the remaining file codes and functions. It is a good way to go through the different functions and their call graphs to get the overview of the code and its working.

If you have encountered any error, then please check if you have setup the CDR database properly and specified the connection information for all the databases correctly in the configuration file. The call type names are case sensitive, so be careful while typing them in the configuration file and also indexing them for CDR database.

References

[1] Sengar, Hemant and Wang, Haining and Wijesekera, Duminda and Jajodia, Sushil: *Detecting VoIP Floods Using the Hellinger Distance*, IEEE Trans. Parallel Distrib. Syst., 2008.

[2] Van Jacobson, Michael J. Karels: *Congestion Avoidance and Control,* Proc. ACM SIGCOMM, November, 1988

[3] The Hobbit Monitor: http://hobbitmon.sourceforge.net/