Constraint-based Trend Template for Intrusion Detection

Md. Ahsan Habib
Software Engineer, iSoftel Co. Ltd., Bangkok, Thailand
Email: ahsan.sust@gmail.com

Krishna Prasad Paudel
Computer Science and Engineering, Kathmandu University, Nepal
Email: pkrsar@yahoo.com
Mohammad Rajiullah
School of Engineering and Computer Science (SECS), Independent University, Bangladesh (IUB)
Email: rajiullah@secs.iub.edu.bd
Prashanta Man Shrestha
Software Engineer, iSoftel Co. Ltd., Bangkok, Thailand.
Email: pmshrestha@gmail.com

Abstract— **Intrusion detection systems (IDS) are special computer security tools which help detect intrusion attempts. Misuse based detection is one of the techniques which is used by IDS to recognize predefined attack signatures. Attack languages, also known as detection languages, are used to describe attack signatures. Detection languages should be simple, expressive and flexible enough to help encode event signature accurately and conveniently. This paper shows the effectiveness of constraint based Trend Template (TT) as an efficient detection language by encoding some attack scenarios and focusing on the Trend Detector which recognizes those signatures from intrusion data.**

Index Terms—**Intrusion detection system (IDS), Trend Template (TT), Trend Detector, Snort, DARPA**

I. Introduction

Computers connected to network are subjected to various kinds of vulnerabilities. The services provided by a networked computer could be exploited by an attacker to gain control over it. Intrusion attempts leave traces to the system under attack. These traces can be tracked through different sources i.e. network packets, audit trails and service logs. Special types of software which tracks intrusion traces are known as Intrusion Detection System (IDS). There are two methods, explored so far, to detect intrusion events in a system – anomaly detection and signature recognition. The second method is required to have database of known attack patterns. Intrusion events are matched against these patterns and known attacks along with their variants are recognized. Attack detection language facilitates describe attack signatures. Till to-date, several attack detection languages exist with their own advantages and limitations.

Ref. [1] is the ground work of this paper. In that paper, trend template concept for encoding intrusion pattern was discussed. This paper explores further by providing attack signatures, detection algorithm, performance evaluation and comparison of the trend detector.

Following sections of this paper are organized as follows: section 2 introduces in detail about theoretical background and overview of modifications been done, section 3 describes two TCP/IP based attack signatures and scenarios, section 4 and 5 shows performance of trend detector and its comparison, last two sections conclude with related works and conclusion.

II. Background and Overview

A. Regression based TT

According to Haimowitz [3], a trend template represents a pattern of multivariate data shared by a diagnostic population in a particular clinical context. The representation combines constraints in both value and time. The value constraints limit functions of measured parameters to fit statistical models expressed as functions of time. The time constraints limit the extent of the value constraints to the appropriate time intervals, each having uncertain endpoints and duration. Together these constraints provide more discriminatory power between different diagnoses than either could provide alone.

Key constructs of trend templates are landmark-times, temporal-intervals, temporal-relations, state-constraints and regression-functions. These key elements are as follows:

- Landmark Times: Represents significant properties of events over time e.g., boundaries between different phases of the event. These could be concrete time (calendar time); or abstract times

Based on “Effective event description using trend template language and efficient intrusion detection”, by Md. Ahsan Habib, Phan Minh Dung which appeared in Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on 7-10 Oct. 2007 on page(s): 326-331 in Montreal, Quebec. © 2007 IEEE.

representing uncertainty relationship between phases of events.

- *Temporal Intervals*: Intervals represent periods of the process that characterize significant sub events. Intervals can bear specific or abstract durations constrained by relations to other intervals and to landmark times.
- *Temporal Relations*: These relations provide shorthand means for expressing relations among intervals that one could express at greater length using the (min max) relations.
- *State Constraints*: These specify characteristics of objects during temporal intervals, such as constant values, increasing or decreasing values, shapes of curves etc.
- *Regression Functions*: They model criteria for matching templates against data, and so describe means for deciding when events occur when uncertainty exists about starting and ending times.

Fig. 1 depicts a trend template which consists of two intervals where the first interval checks for a steady BP (blood pressure) followed by a linear decrease of BP in the 2nd interval. Through x-axis scan time and y-axis interval parameter value (BP in this case) are measured. Whole pattern is checked in process data for five minutes. Points, *Now* and *-5 min* are two points indicating start and end points of the template.

The meeting point of two intervals can vary from 1 to 5 minutes. Negative minute values indicate past in time compared to current time stamp *Now*.

B. Trend Detector for Regression based TT

In model based trend detection, each interval is attached with one or more value constraints. This constraint language consists mainly of linear and quadratic regression models for numeric data. Assignment of process data against a trend template was subjected to temporal context which was merely representation of different variants from the exact regression model. Each different temporal context gave birth of new hypothesis. These hypotheses are the active elements which are matched against the predefined statistical model defined in TT. There could be more than one hypothesis attached with each template during the matching process. As no absolute match could be determined, a group of top hypotheses with higher match scores were maintained. The trend matching algorithms rely on the Temporal Utility Package (TUP) [4] that propagates temporal bound inferences among related points and intervals.

C. Constraint based TT in Intrusion Detection Context

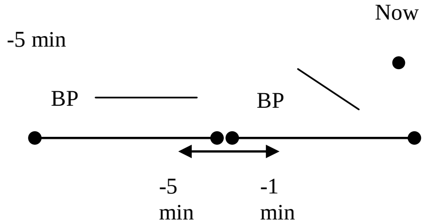


Figure 1. Regression-based Trend template for a recent decreasing blood pressure.

BP example in section II (A) displays that the interval value constraints are expressed as statistical models (linear increase, steady, exponential decrease and the like) which provide abstraction in describing patterns. This kind of description matches the description provided by the medical experts while describing the medical history of a patient. Constraint based trend template is a variant of original regression based trend template where the interval value constraints are expressed using range of values. Each value constraint is an expression of the form

$$m \leq f(D) \leq M$$

where f is some real valued function defined on a set of process data, m is a minimum, and M is a maximum.

Regression based trend template and partial matching based detection algorithm was found appropriate in event description in biological context but for intrusion detection context, constraint based trend template was found more appropriate. According to general constructs of trend template, landmark points exist which will bind the boundaries of different intervals with the process time. Also the start and end points of a trend template are two important landmark points which indicates the exact temporal position of starting and ending point of trend template in process data. Special landmark point, called *Now,* is used that denotes the time of the currently processed datum. Fig. 2 demonstrates an attack signature which can be characterized by two consecutive stages.

Before describing mechanism of this trend template, basic constructs need to be described. Two intervals, interval-1 and interval-2, each with their own value constraints are constraining some parameters in process data (in vertical direction) where as the horizontal axis indicates scan time in seconds. The template holds the signature with length only 5 seconds. Two landmark points, *-5 sec* and *now*, indicate the start and end points of the template. Interval constraints express a parameter in (min, max) range i.e. in interval-1 (min, max) is (a, A) where a<=A and in interval-2 (min, max) is (b, B) where b<=B.

Total scan length is mentioned 5 seconds and this time is attack specific. Each interval consists of one value constraint. For example, Interval-1 has value constraint which indicates that the concerned parameter(s) need to stay between minimum and maximum value a, and A respectively. The transition point between the phases is between one and five seconds before landmark *Now*. The duration variance is attack specific. This flexible feature enables intervals to expand or collapse temporally as required.

D. Inconvenience in Encoding Attack Signature

Still some inconveniences exist with constraint based

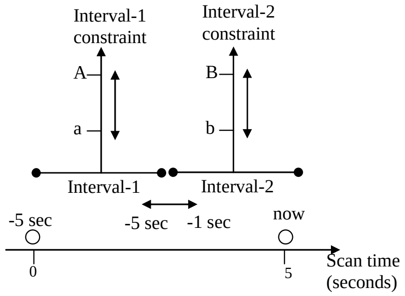


Figure 2. Sketch of a constraint-based trend template

template in figure-2 in describing attack signature and which are the so called start, and end landmark points. The strictness of mentioning temporal distance between two landmark points is also a negative point. The fact is that trend template concept was introduced to encode medical trends of human being where human *birth* and *death* were considered as the start and end landmark points of a trend template. Also it is customary to express the temporal distance of an intermediate landmark point with reference to the *start* and *end* landmarks. But in the case of intrusion detection, things are different which is discussed in next section.

E. Improvements of Constraint-based Trend Template for Encoding Attacks

In the context of clinical trend detection, for designing a trend template of a disease pattern, two landmark points, start and end, were always used. It was meaningful because the process data which was monitored to detect desired pattern was nothing but the medical history of human being and each human being has birth and death which match exactly with start and end landmarks of the template. For detection mechanism it was designed that way. But for network traffic which we consider as process data doesn't have start and/or end points. Network traffic is a continuous source of data. Two landmark points, start and end, have no significance in the context of intrusion detection. By default, start point of a trend template can be bind to the process data as detection starts but the end point can not be predicted. If IDS program is running in a server which is live for months, then the end point of the template also need to be proportional with that.

For example, what is the temporal distance between start and end point in detecting nmap probing is unpredictable beforehand. If we relax start and end landmark points, then templates become more expressive and effective in terms of encoding attack signatures. Fig. 3 is shown below with the modification of figure 2.

In figure 3, start and end landmark points were removed. More specifically, two landmark points from figure 2, *-5 sec* and *Now*, were removed with other things remaining unaffected. Now the limitations mentioned in previous section can be addressed as below by mentioning the added flexibilities –

(a) No need to specify start and end points in describing trend template though start point of template is by default the begin of the first interval.

(b) As landmark points are not significant any more, temporal uncertainty of start/end points of intervals can be represented using only start/end points of other intervals or it can be absolute time in hours, minutes, or seconds same as original concept. For example, consider the constraint: begin of 2nd interval should not delay more than one second after the end point of 1st interval. This is easily achievable according to current trend template structure.

(c) There is no need to mention the temporal length of a trend template. Rather this temporal extent is dependent on the individual interval which can grow/shrink according to their need. This is very important for intrusion detection context. With even *Now* based representation as shown in figure-2, temporal length needed to mention explicitly which has been found impractical in intrusion detection context and thus have been removed.

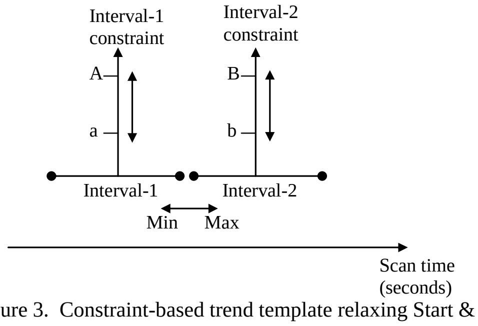


Figure 3. Constraint-based trend template relaxing Start & End landmark points

F. Trend Detector for Constraint-based TT

An algorithm for detecting regression-based trends has been developed by Haimowitz [3] but we need one which will detect constraint-based trends rather regression-based. One such algorithm is developed and discussed below. Processing of data is done forward in time, reasoning simultaneously about a cluster of data that are measured at the same time slice for the same process. For example, network packet data and secure-log data of same time slice can be considered as a single cluster of data for a 'Dictionary Telnet' attack. Each data cluster is matched to all templates that constrain some of the parameters of the data in that cluster. For each such data cluster Dt, where t is the time of data in the cluster, trend detector evaluates TREND-DETECTION [Dt], detailed below.

1) Trend Detection Algorithm

The detection algorithm is described below. It is followed by description of two of its functions Update-TT-Interval-Status, and Get-Current-Active-Interval

```
TREND-DETECTION (Dt: network data or data-
cluster)
  FOR all template TT network data Dt is feed as
      input source
    Let currentInterval = Get-Current-Active-
      -Interval (TT);
    IF currentInterval is positive AND parameter is
        matched with Dt
      Update all constraints in currentInterval with new
        data Dt accordingly;
      Update-TT-Interval-Status(TT, currentInterval);
      IF currentInterval is the last interval AND whose
          value constraints are TRUE
        Declare and Log detection success of TT
        Reinitialize all data-structure and Restart scan
          for TT;
      END IF
    END IF
  END FOR TT
END TREND-DETECTION
```

```
Update-TT-Interval-Status ( TrendTemplate TT, TT-Interval curIval)
  BOOLEAN constraintsSatisfied = TRUE
  FOR all constraints cons IN curIval
    IF cons IS FALSE
      constraintsSatisfied = FALSE;
    END IF
  END FOR cons
  IF constraintsSatisfied IS TRUE
    SET curIval inactive
    IF (curIval+1) EXIST
      SET (curIval+1) active
    END IF
END Update-TT-Interval-Status
```

```
Get-Current-Active-Interval ( TrendTemplate TT)
  FOR all intervals in TT
    IF any active interval found
      RETURN the index of that interval
    ELSE
      Return negative integer
    END IF
  END FOR
END Get-Current-Active-Interval
```

G. Detection Mechanism of Trend Detector

Trend Detector is an engine which implements the trend templates and searches for *patterns* in the process data according to TT specification. Data cluster Dt is feed as input stream. This data could be network traffic or audit trails or both. Variable *currentInterval* stores the index of current active interval from the function Get-Current-Active-Interval. At any particular instance of time only one interval in a multiple interval template will stay active whereas all other intervals will be inactive. The function Get-Current-Active-Interval takes care of that.

If the data Dt matches with parameter(s) of interval of a template, then the internal data structure, which keeps record of necessary information, will be updated. Immediately after this step, checking is performed if constraints of *currentInterval* are satisfied or not. Function Update-TT-Interval-Status takes care of it. If all constraints become true, then current interval is deactivated and the next interval, if any, is activated. The steps repeated again by feeding Dt to the detector and in this way detection forwards with TT intervals in time. The value of *currentInterval* is negative if all the intervals of template are inactive. This situation indicates that the last interval's value constraint was satisfied and that's why it is deactivated. It is the case when we can say that the characteristics specifying the trend template, TT, was detected. This information is logged to a file or announced to the console.

Consider the two cases upon finding of which the data structures are reset and scan proceed from beginning. First, it happens when all intervals of the template TT is matched with the process data Dt and second, when any intermediate interval exceeds the maximum temporal extent constraint with its previous intervals. If this intermediate interval is the last interval then, it indicates that the template TT was not matched with the process data and further delay will violate temporal constraint.

The flow chart of detection mechanism is shown in figure 5 which gives a pictorial description of the trend detection mechanism.

III. Attack Specification

This section describes the nature of two IP based attacks, graphical representation of corresponding Trend Template, and formal representation using XML format which is understandable by trend detection module. XML representation of only ICMP Smurf will be included in appendix A.

A. ICMP Smurf

When an attacker sends an ICMP echo request to broadcast address of a network by forging the source address to a victim machine IP address, then ICMP echo responses coming from all the hosts of that network. The nature of an ICMP Smurf attack can be summarized as follows –

- Presence of multiple source addresses from where icmp echo-reply are coming. It is customary that hosts from multiple networks could be involved.
- All icmp echo-reply flood are coming for a single destination host

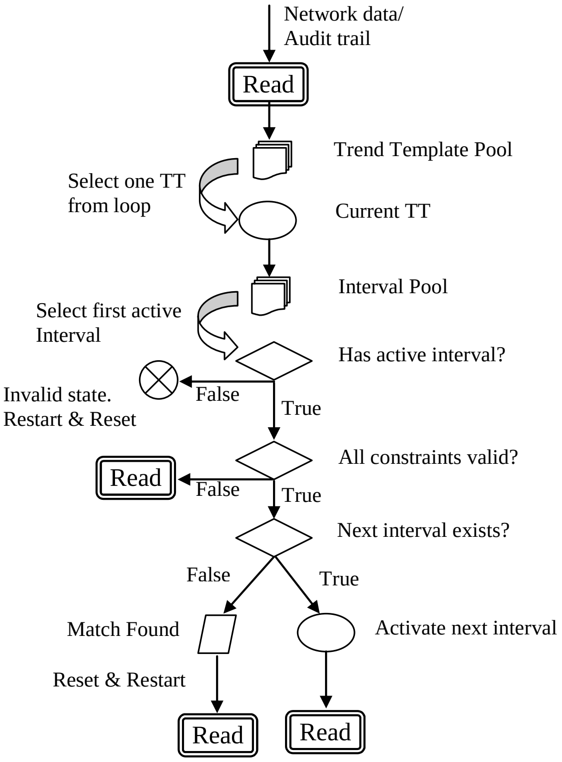


Figure 4. Trend template of ICMP-Smurf

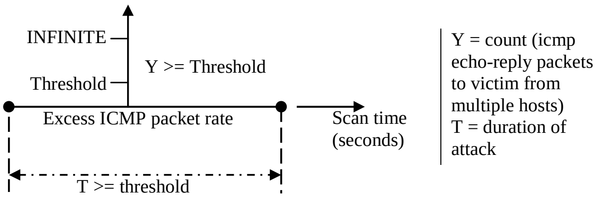


Figure 5. Trend template of ICMP-Smurf

- Network packet category is ICMP echo-reply.
- Generally, ICMP packes have both id number and sequence number set to zero. But it could be varied based on tools used to launch the attack.
- The packet flooding duration should be more than 3 seconds. Otherwise, it could be assumed as a discrete event rather than an ICMP Smurf.

Top three of those five points are the essential characterizing features of ICMP Smurf attack. The fourth one was observed both in DARPA intrusion dataset and in simulated intrusion data, so, it was incorporated as one of the characterizing features for this attack. The fifth one, attack duration, is set to a minimum value. In DARPA dataset, the ICMP Smurf duration was set to at least 3 seconds. Three minimal essential feature subset mentioned below could be set as detection criteria:

- Feature-1: Multiple source addresses
- Feature-2: To single destination address (victim machine)
- Feature-3: Packet type is ICMP echo-reply

Iff all three features are encountered in network traffic; it is an ICMP Smurf attack. Let's have a close observation to the trend template in fugure-5 if it could encode the detection pattern of ICMP Smurf.

In other words, it is required to check if the above figure says at least about all those three features of icmp-smurf attack. It is seen that there is one interval, Excess-ICMP-packet-rate, with one value constraint and one temporal constraint. The parameter is Y, which is –

Count(icmp echo-reply packets to victim from multiple hosts)

It is a function which counts the number of packets which meets some criteria mentioned by the literal. The above literal could be broken into the following sub-strings –

<u>icmp echo-reply</u> packets <u>to victim</u> <u>from multiple hosts</u>

Above underlined substrings: *icmp echo-reply*, *to victim*, and *from multiple hosts* send message to the intrusion detection system to take care about the three key features from minimum feature-set which are stated according to the order of the substrings –

- ➢ Packet type is icmp echo-reply
- ➢ Packets being send to single destination host (victim host)
- ➢ From multiple source addresses

As the trend template was found to successfully represent the above three messages, it is clear that it is containing icmp-smurf attack signature.

B. Port Scan (using Nmap)

This section describes the issues related to nmap probing. Nmap is a well known network analyzing tool with lots of options to scan a network or single host to get different kind of information. It is used to enquiry as simple as, if a host is alive or to more complex, the operating system version or different service information running on a particular host. Let's observe in figure-6 the intrusion traffic of nmap scan where one source host is interested in doing a SYN-scan to a destination host. Linux based hosts were used for generating traffic and launching nmap. Nmap for windows might have different scan pattern.

Figure-6 reveals the characteristics of SYN-scan. The source 203.159.18.147 scans for open ports in destination host 203.159.46.66. First an ICMP echo request is sent which is followed by an ACK packet to the port number 80 (port name: http). These two packets are the starting for almost all variety of nmap scans. It is certainly not clear at this point what type of scan it is, but it is clear that an nmap scan has been launched. The time gap between these two packets should not exceed few hundred milliseconds. For further enquiry about the type of scans, the third packet needs to be observed. If it is a tcp type scan (SYN-scan, FIN-scan, ACK-scan etc.), the tcp-flag bit of third packet will reveal the scan type. In this scan, it is seen that a SYN packet is sent to the port named ***auth***, which clearly says that it is SYN-scan. It should be noted that it is not always evident that the 3rd packet always correctly says about scan type. For some scan variety, the deviation from this assumption was observed.

The following are the features which can be used as signature for nmap scan –

- Packets coming from same port of a source machine
- Packets are for single destination
- First packet is ICMP echo-request followed by an ACK packet to the port number 80. The time gap between these two packets should not exceed few

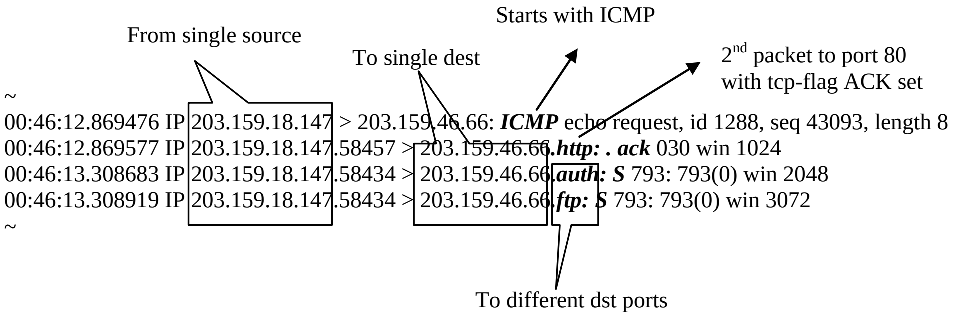

```
~
00:46:12.869476 IP 203.159.18.147 > 203.159.46.66: ICMP echo request, id 1288, seq 43093, length 8
00:46:12.869577 IP 203.159.18.147.58457 > 203.159.46.66.http: . ack 030 win 1024
00:46:13.308683 IP 203.159.18.147.58434 > 203.159.46.66.auth: S 793: 793(0) win 2048
00:46:13.308919 IP 203.159.18.147.58434 > 203.159.46.66.ftp: S 793: 793(0) win 3072
~
```

Figure 6. Nmap attack signature in intrusion traffic

hundred milliseconds.

- The third packet says about the scan type of nmap scan (though always not correct)

The top three among four features ensure the existence of nmap scan. Figure-7 depicts how the above mentioned features are encoded in trend template.

The figure consists of two intervals as we are interested to detect only two packets – one ICMP followed by one ACK at port 80 in no later than 200 milliseconds. It was observed from figure 6 that the difference of time-stamps for these two packets was around 100 milliseconds. To allow more flexibility, this gap requirement was doubled. Due to bandwidth of the network, this time gap could vary. The parameter for first interval, detect icmp, is –

count (*ICMP echo-request* to victim)

where the sub-string of interest is highlighted.

And, the parameter for second interval, probe at port-80, is –

count (packet *from same source port* to *dst port 80* & with *tcp-flag ACK*)

where the sub-strings of interest are underlined. These underlined sub-strings ensure about the three characterizing features of nmap scan except the minimum time gap between first two packets. This option is mentioned in the figure rather.

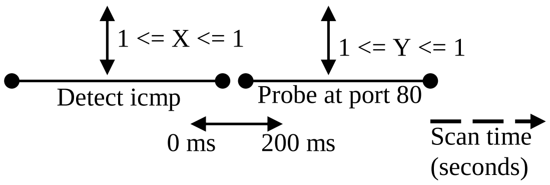
IV. Performance of trend Detector

Performance of trend detector is discussed for traffic generated by attack simulation tools and DARPA 1998 intrusion datasets as well. Five different attacks were simulated using automated tools and they are icmp-smurf, ping-flood, syn-flood, dictionary-telnet, and nmap-probe. But the DARPA attack taxonomy didn't include all these five attacks in the intrusion dataset. To evaluate performance against DARPA intrusion dataset, three attacks were chosen – named as nmap, smurf-icmp, and ipsweep. The following sections describe performance against dataset of both the categories.

A. Performance for simulated attacks

As network traffic of simulated attacks was studied closely, generated templates were more accurate. Simulated attacks were launched with different options extensively and trend detector detected them successfully. The detection status is shown in the table 1.

These intrusion generating tools are either executable program written in C or shell/perl script which generates intrusion data. Table 1 shows qualitative results rather than quantitative where the status only shows if those attacks were detected by the detector or not.



⇒ X = count (ICMP echo-request to victim)

⇒ Y = count (packet from same source port to dst port 80 & with tcp-flag ACK)

⇒ *Interval Gap not more than 200 m*

Figure 7. Trend template of Nmap probe

B. Performance for DARPA Listed Attacks

Both seven weeks of training data and two weeks of test data contain attacks in the midst of normal traffic. Here results are included for test dataset only. No attack list file is provided with test dataset. One alternative list was found from Kendall's thesis [5]. Rather considering the intrusion data of every day in two week duration, concentration was provided on data relevant to chosen three attacks icmp-smurf, ipsweep, and nmap probe.

Ref. [5] provides results for DARPA test dataset in such a way that it was found inconvenient to count the number of attack instances. The false positive and false negatives were measured in terms of source/destination hosts/networks. Table 2 shows that detection results. It is observed that detector missed smurf detection in week-1 Tuesday for 97 subnets. Among involved sources from 100 subnets, sources from only 3 subnets were detected – it's huge performance degradation. It was observed that the intrusion data file was corrupted as Tcpdump failed to read up to the end of the file. The trend detector detects smurf well for data other than the day of error.

The trend detector also detects nmap probe well. Same tcpdump read error occurred for nmap scan at week-2 Thursday where one source/destination pair was not detected. As the scan could not continue till end of the intrusion data file, this performance gap was considered positively.

Ipsweep could not be detected well in test dataset. It is seen that it missed one scan in week-2 Monday. But it detected another ipsweep instance in week-2 Friday. But that detection came with a cost of false positive for nmap probe. Icmp packets of Ipsweep probe looks same as nmap's icmp packet and it was found difficult to differentiate.

V. performance Comparison between Trend Detector and Snort

Snort [6] is a popular network based IDS. Its detection mechanism is per packet basis. Each rule that contains a specific pattern is matched against individual packet. Each rule in the rule chain is verified recursively and if a match is found, it is announced and if not, then simply the next rule is checked. This process is continued unless all the rules in all the rule chains are ended up. Each rule is mutually exclusive and stateless as well. Snort manages to overcome such limitations i.e. statelessness by using preprocessors. Roughly speaking, for each class of

Table 1. Detection of simulated attacks

Attack	Detection Status
ICMP-Smurf	Detected
Ping Flood	Detected
Syn-Flood	Detected
Dictionary-Telnet	Detected
Nmap Probing	Detected

Table 2. Detection of DARPA attacks from test dataset

Week	Day	Attack name	Original Source/Dstination	Detected Source/Dstination	False Positive	False Negative
1	Tues	Smurf	100 subs/172.16.114.50	3 subs/ 172.16.114.50	0	97 subs
	Thurs	Nmap	207.253.84.13/172.16.118.20	207.253.84.13/172.16.118.20	0	0
		Smurf	10 subs/172.16.118.30	8 subs/172.16.118.30	0	2
	Wed	Nmap	207.253.84.13/172.16.118.20	207.253.84.13/172.16.118.20	0	0
		Smurf	199.174.194/172.16.114.50	199.174.194/172.16.114.50	0	0
2	Mon	Smurf	10 subs/172.16.112.50	10 subs/172.16.112.50	0	0
		Smurf	All attack /172.16.114.50	7277 hosts/172.16.114.50	?	?
		Smurf	152.169.215, 202.77.162 /172.16.114.50	152.169.215, 202.77.162 /172.16.114.50	0	0
		Smurf	255.255.255.255, .el .al /172.16.112.50	12.2.2, 1.1.1 /172.16.112.50	?	?
		Ipsweep	204.97.153.43/*	0	0	204.97.153.43/*
	Tues	Smurf	209.1.12/172.16.118.20	209.1.12/172.16.118.20	0	0
	Wed	Nmap	207.253.84.13/172.16.118.20	207.253.84.13/172.16.118.20	0	0
	Thurs	Nmap	207.253.84.13, 128.223.199.68 /172.16.118.20, 172.16.113.50	207.253.84.13, /172.16.118.20	0	128.223.199.68 /172.16.113.50
	Fri	Ipsweep	135.13.216.191/172.16.114	135.13.216.191/172.16.114	0	0
		Nmap	0	135.13.216.191/172.16.114	135.13.216.191 /172.16.114	0

attacks, it uses one preprocessor i.e. '*sfPortscan*' preprocessor is responsible for detecting different kinds of port scanning. The '*Stream4*' module provides TCP stream reassembly and state-full analysis capabilities to Snort.

Intrusion traffic was simulated using automated tools. Attacks in table-3 were launched while both snort and trend-detector were running. This process was repeated several times to check the consistency between results. Snort version 2.6.0 was considered for testing.

Snort rules are not capable for post-processing packets and detect the multiple sources' involvement (smurf feature) in ICMP-Smurf attack but it detects individual ICMP packets. For this reason, snort performance is considered partial. Snort's nmap probe detection rate is sufficient. Unfortunately, snort generates no alert for *dictionary telnet* attack. May be snort assumes no threat for excessive number of telnet connections to the server though there were false login attempts. Snort probably has no option to scan security log file to check false login rate as a host-based IDS does.

VI. Related work

Haimowitz [3] introduced trend templates in his PhD dissertation and showed its effectiveness to describe biological patterns in historical medical dataset in more abstract way along with a detector, TrenDx.

Table- 3: Comparative performance between Snort and Trend Detector

Attack	Trend detection	Snort
Icmp-smurf	Detected	Partial detection (smurf blind)
Syn-flood	Detected	Partial detection
Dictionary-telnet	Detected	Partial detection
Nmap (xmas+few varients)	Detected	Detected

Doyle et al. [2] mentioned in his article about the issue that trend templates could be used in describing events signature more effectively. Comparison with other event description languages was shown in that article. They tried to prove that trend templates have potential to express events more effectively which would help recognize events more effectively than traditional intrusion detection methods i.e. anomaly detection and misuse detection. This paper was an attempt to identify practical issues that should be incorporated in TT concept in order to use it in intrusion detection context.

VII. Conclusion

Combination of host and network based IDS could be used to provide enhanced security features since each IDS performs well for different features. The issues explored so far about trend template shows possibilities to get more from this approach. Encoding more attacks and improving the detection technique is a future goal.

Appendix A icmp-smurf XML template

The XML representation is shown below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<trend_template name="icmp_smurf">
<intervals>
<interval name="ExcessIcmpPacketRate">
<constraints>
<constraint type="value">
<function>
<type unit="second">rate</type>
<range>
<min>5</min>
<max>INFINITE</max>
</range>
</function>
<params>
<protocol type="echo reply">icmp</protocol>
```

```
<source_ip>*</source_ip>
<dest_ip> a.b.c.d </dest_ip>
</params>
</constraint>
<constraint type="temporal">
<function>
<type unit="second">duration</type>
<range>
<min>3</min>
<max>INFINITE</max>
</range>
</function>
</constraint>
</constraints>
<next_interval>NULL</next_interval>
</interval>
</trend_template>
```

REFERENCES

[1] Md. Ahsan Habib, and Phan Minh Dung, “Effective Event Description Using Trend Template Language and Efficient Intrusion Detection” in IEEE International Conference on Systems, Man and Cybernetics, Oct. 2007, Montreal, QC, Canada, P: 326-331.

[2] Doyle J., Kohane I., Long W., Shrobe H., Szolovits P. (2001). Event Recognition Beyond Signature and Anomaly. In *Proceedings of the 2nd IEEE-SMC Information Assurance Workshop (2001), 17—33*

[3] Haimowitz I. J. (1994). *Knowledge-Based Trend Detection and Diagnosis*. (Doctoral dissertation, Department of Electrical Engineering and Computer Science on May 12, 1994). Massachusetts Institute of Technology.

[4] I. S. Kohane, “Temporal reasoning in medical expert systems,” in MEDINFO 86 : Proceedings of the Fifth Conference on Medical Informatics, R. Salamon, B. Blum, and M. Jorgensen, Eds., Washington, Oct. 1986, pp. 170-174, North-Holland.

[5] Kendall, k. (1999). *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. (Master Thesis, MIT, 1999). Massachusetts Institute of Technology.

[6] Snort documentation page. http://www.snort.org/docs/snort_htmanuals/ htmanual_261/

Md. Ahsan Habib was born in Bangladesh in 1979. He received his Master degree in information and communications technologies from Asian Institute of Technologies (AIT), Bangkok, Thailand in 2007 and his Engineering degree in computer science and engineering from Shahjalal University of Science and Technology (SUST), Sylhet, Bangladesh in 2003.

He is currently a Senior Software Engineer in iSoftel Co. Ltd., Bangkok, Thailand. Previously he was working as a lecturer in computer science & engineering in Asian University of Bangladesh, Dhaka, Bangladesh. He published conference paper which is mentioned in Ref. 1. His research interest includes secure application design, and software engineering.

Mr. Habib is a member of IEEE since 2007.

Krishna Prasad Paudel was born in Nepal in 1970. He received his Master degree in information and communications technologies from Asian Institute of Technologies (AIT), Bangkok, Thailand in 2007 and his Engineering degree in electrical and electronics engineering from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh in 1995.

He is currently a lecturer in department of Computer Science and Engineering in Kathmandu University, Nepal. His research interest includes compiler design, web technology, and network security.

Mohammad Rajiullah was born in Bangladesh in 1981. He received his Master degree in global information and telecommunication studies from Waseda University, Japan in 2007 and his B.Sc. degree in computer science & information technology from Islamic University of Technology (IUT) Dhaka, Bangladesh in 2002.

He is currently a lecturer in School of Engineering and Computer Science (SECS) in Independent University, Bangladesh (IUB), Dhaka, Bangladesh. His research interest includes ubiquitous computing, artificial intelligence, and computer security.

Prashanta Man Shrestha was born in Nepal in 1982. He received his Master degree in mechatronics from Asian Institute of Technologies (AIT), Bangkok, Thailand in 2007 and his Engineering degree in electrical and electronics engineering specialized in communication from Kathmandu University, Nepal.

He is currently a Senior Software Engineer in iSoftel Co. Ltd., Bangkok, Thailand. His research interest includes machine vision, embedded system, user centered design, and network monitoring system.