computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

TOWARDS DATA MINING TEMPORAL PATTERNS FOR ANOMALY INTRUSION DETECTION SYSTEMS

Sam Sengupta [1], Bruno Andriamanalimanana [1], Stuart W. Card [2], Pradnya Kadam [1], Saket Ranwadkar [1], Kaustav Das [1], Sagar Parikh [1]

1) State University of New York Institute of Technology, Utica NY 13504-3050,
sengupta@sunyit.edu, fbra@sunyit.edu, kadamp@sunyit.edu,
ranwads@sunyit.edu, dask@sunyit.edu, parikhs@sunyit.edu
2) Critical Technologies Inc., 1001 Broad Street - Suite 400, Utica NY 13501,
stuart.card@critical.com

Abstract: *A reasonably light-weight host and net-centric Network IDS architecture model is indicated. The model is anomaly based on a state-driven notion of "anomaly". Therefore, the relevant distribution function need not remain constant; it could migrate from states to states without any a priori warning so long as its residency time at a next steady state is sufficiently long to make valid observations there. Only those intrusion events (basically DOS and DDOS variety) capable of triggering anomalous streams of attacks/response both near and/or far of target monitoring point(s) are considered at the first level of detection. At the next level of detection, the filtered states could be fine-combed in a batch mode to mine unacceptable strings of commands or known attack signatures.*

Keywords: - *IDS, Anomaly detection, DOS, DDOS, NIDS*

1. INTRODUCTION

One major problem of the current century that seriously threatens the emerging Information Technology industry must be our increasing inability to comprehensively deal with the issue of Information Assurance in cyberspace -- the ubiquitous Internet or private internets both in the civilian and in the government corridors. The attackers of yesteryear return with ever new approaches, foiling attempts to protect critical information resources; sometimes they are apprehended but the damage to the industry, to economy, to the protected information base continues unabated.

This paper is about intrusion detection [1,2,3] in a network system supported by anomaly analysis of network traffic as attempted by SPADE [7] and SPICE [8]. Anomalous traffic is identified as a potential intrusion event; its detection doesn't depend on knowledge of attack signatures and therefore it is best suited to detect new attacks for which signatures have yet to be developed. Our model hinges on online statistical analysis of actual packet traffic as observed by promiscuous capture (ala tcpdump) or MIB variables at a host (local traffic), at a router (neighborhood traffic), or as logged events of dropped incoming traffic at firewalls, or buffer overflows observed at an interface, etc.

We assume that an intruder's reconnaissance of the system S(t) may generate at times an avalanche of traffic at a site or in the neighborhood of a router either as a result of persistent attacks or as the system's response, causing, in some sense, a statistically anomalous traffic pattern. In our paper, we indicate how such intrusion events may be detected online, how, from the list of saved states at and before intrusion events, observation of significant correlation among seemingly independent intrusion event patterns may trigger alarms for the future, and how all such approaches could be integrated into a coherent base on which both known and unknown intrusion events could be captured, if possible, before damage is done.

Minimally, an Intrusion Detection System (NIDS at the network level, or IDS at the host or a router level) in our design is conceptualized as a localized entity as shown in Fig 1. It is a recursive architecture in the sense that at any stage it has to identify any anomaly from the set of raw data in its current buffer and post such findings with an alert at the next level down

The idea is that some anomalous events E remain anomalous even when the target system as a whole experiences a concept drift; these must be identified

online, and if they somehow escape at level l as legitimate traffic, the anomaly could potentially be discerned at the next level $l+1$, and so on. Expanding such architecture to include the entire internet in order to monitor resource usage in a broader neighborhood would accord us an even better grip on intrusion events initiated elsewhere, and a manifest for a potentially global reach. This would be the approach Homeland Security could undertake since monitoring any private network segment or traffic is otherwise illegal.

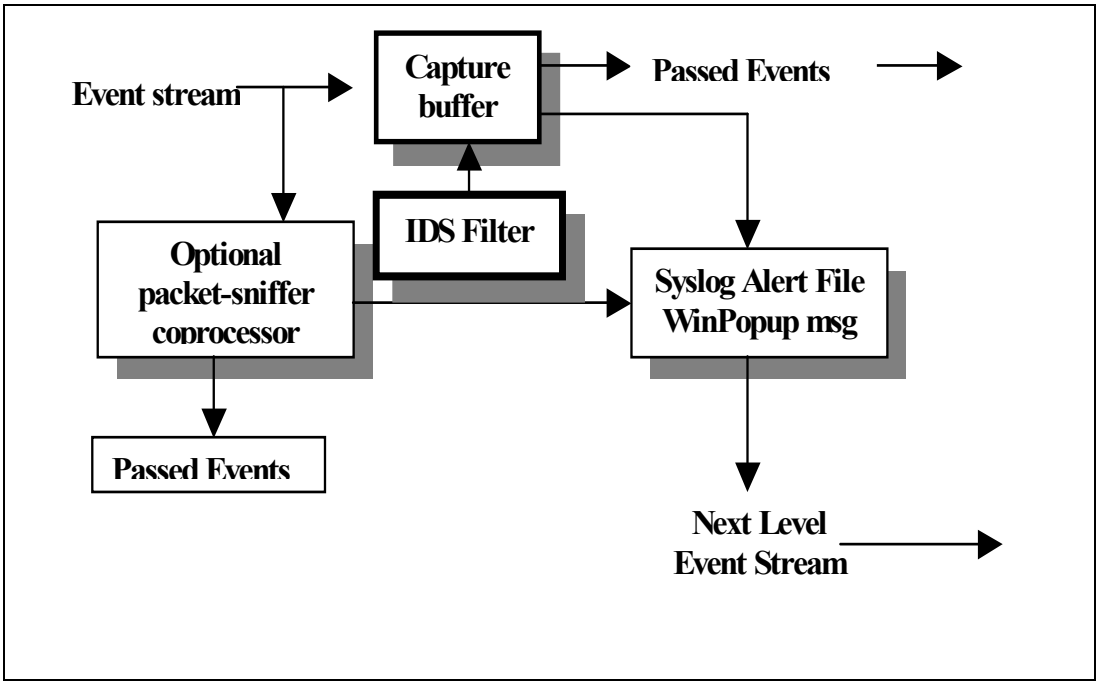

Fig. 1 - IDS Architecture.

Note that a packet-sniffer coprocessor in the proposed architecture complements the IDS filter in order to identify attacks employing header modification. Such modifications, seen as errors by the packet-sniffer, would not always lead to alert-events that are expected to be captured by the IDS filter; they might escape inspection there. The coprocessor could be used as an optional feature in parallel with the IDS filter and, since event postings at the Syslog file must be on a real-time basis, we require the alert detection coprocessor to respond to alert-events with minimal delay. Since it is optional, one could also expand the coprocessor to include a multiple set of concurrent coprocessors each of which could act as an independent rule-server checking out possible violations to one or a few rules the packet must obey.

Advances in storage technology allow for the capture of significant of amounts of data, much of which is not necessarily information to the IDS. Our approach is to extend memory management with compression and selection techniques that mirror the hierarchical nature of the IDS process. An engine controls each (compression and selection) stage by monitoring its own dataflow, feeding forward the interesting data and responding to feedback from the lower engines. Figure 2 illustrates a three-stage compression and selection approach for IP packet, header and flow processing. At each level, the memory allocation is scaled to meet the inflow rate and classification window requirement.

The selection of a mid-buffer trigger point affords the IDS classification filter future and past information that might avert unnecessary processing that is typically associated with no-memory or history-only packet analysis. Note that 'capture' really refers to 'retention': all packets are briefly held in the first stage buffer; if they satisfy a trigger condition, they are kept; if not, payload statistics are updated, payloads discarded, and headers move on to the next stage. Retention is temporary for uninteresting raw data and short-term statistics, to limit storage requirements; but permanent for highly interesting raw data and long term statistics, to support post-mortem analysis.

The nominal TCP three-way handshake (SYN → SYN ACK → ACK) typically would not be flagged as anomalous, if the packet capture memory were large enough to accommodate the stream. If congestion delays prevented timely capture in the packet buffer, each packet would be marked as interesting and would naturally age to the next stage

where, in this case, the data is trimmed and the header remains intact. When a header reaches the buffer trigger point, it now competes with similarly selected candidates. Due to the reduction in storage element size and the initial classification, a larger time interval can be observed for anomalous events, such as unsuccessful connection attempts.

The natural extension is to address micro-flows where connection duration and port usage are of interest. Here even a successful connection can be termed interesting by virtue of its context at this level. Consider a successful

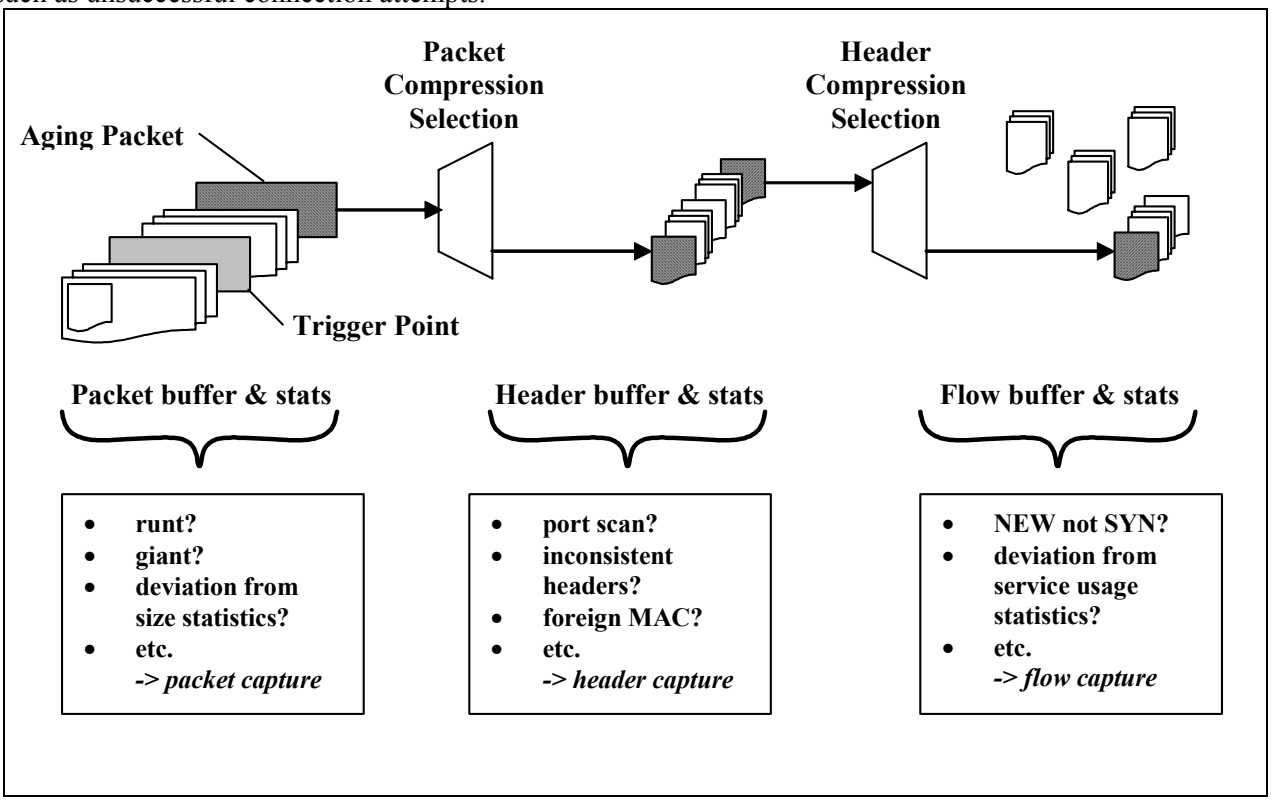

Fig. 2 – Multivariate Multilevel Compression and Selection.

.

FTP login, but no related data connection is made. Certainly a legitimate action, but one that should be flagged as anomalous! Extension may continue to arbitrary levels of macro-flow aggregation with attendant compression of buffered date and consequent lengthening of buffer window temporal extent. The approach is multi-resolution (or multi-scale), non-stationary statistical profiling

2. IDS BASED ON ANOMALOUS CONCEPT DRIFTS

This chapter briefly indicates how intrusion events may conceptually be detected from the network traffic around the system. Assume that a dynamic system $S(t)$, one we want to protect, is in equilibrium or in a state of "quasi-equilibrium" in the sense that it is allowed to migrate slowly from its current equilibrium state φ_e to another neighboring equilibrium state φ_f in a time-interval Δt such that $\left|\frac{\Delta\varphi}{\Delta t}\right| < \tau_{threshold}$. The system is in its "normal" state if either it stays in its last observed equilibrium state for a substantial period of time or is confined to move within a set of equilibrium points so as to be in a "quasi-equilibrium" state as per our requirement. The system S(t), when it transits from a steady state ϑ, undergoes a "concept drift" and is assumed to be in a "roaming" state until it discovers its next equilibrium state ξ.

An intrusion event X(t), in our case, is basically like a port flooding, port walking, probing, online password cracking attempt etc., defined here to be any inexplicable or unauthorized tweaking of the system that forces S(.) to depart from its latest equilibrium norm. This happens, for instance, when the system in question at some sampling time $t + \Delta t$ is not at its latest observed equilibrium state but at some other distant state (shown as a circle) (see Fig. 2), or seeking still another equilibrium state to park but nothing is available in the immediate vicinity but a non-stationary or a roaming state shown as a square in the diagram [5,6].

In figure 3, the system in its phase space moves either from steady states to steady states instantly or

from a steady state to a non-stationary state first and then to a steady state. In the first case, the equilibrium is unstable, but, more importantly, depending on *X(t)*, it could, from its current steady state, migrate to either (a) neighboring steady state , or, (b) to a distant steady state. This is akin to moving from one cluster to either a neighborhood cluster in one jump and stay there or to a distant cluster after a while.

Given our functional definition of "quasi-stationary" state earlier, the event X(t) becomes an intrusion if it leads to a situation in (b). Note that a real intrusion event (in an actual system) may not always force its victim to take giant leaps in phase space. It could manipulate the system to stay within a sequence of legitimate equilibrium state points slowly pushing it out to an undesirable absorbing state for the final kill and yet actual damage would not be known for a while. Obviously, no simple solution could be found to detect such events in terms of our expected "normal" system behavior [5]. Therefore, we define

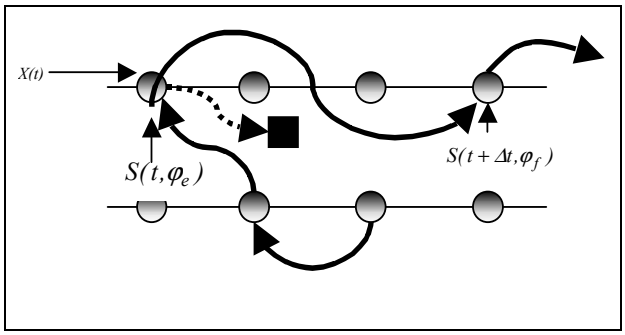

Fig. 3 – System movement in phase space.

Definition 1. An event $e \in \Sigma^*$ is a potential intrusion event if e causes the system to migrate from a steady state ϑ to a roaming state (or a non-equilibrium state) ζ such that $|\vartheta - \zeta| > \tau_{\text{threshold}}$. Otherwise, the event $e \in \Sigma$ where $\Sigma \cup \Sigma^* = U$ some universal set , and $\Sigma \cap \Sigma^* = \varnothing$

. (1a)

Definition 2. Every event $e \in \Sigma^*$ causes the following response:

$e \in \Sigma^* \Rightarrow trigger(alarm, SB(e, \tau, (IP, port), protcl))$

: post event in buffer SB online

$SB(e,...) \Rightarrow e \in False\_positive \,||\, e \in Intrusion$ (1b)

For event $e \in \Sigma \Rightarrow release\_event(e, log\ file)$: event is released for normal consumption

$log\_file(e) \Rightarrow (e \in Normal\_event \,||\, e \Rightarrow False\_negative)$ (1c)

Definition 2 asserts how finally an event would appear to our NIDS filter. If an event is considered a potential intrusion, it would trigger an alarm and then it would be saved in the first-level short term buffer SB. Such an event could be either a true intrusion or a false one (a legitimate event by a bonafide user). For other events, they would just be thrown out as garbage, though a portion of them would prove to be intrusion events (or false negatives).

Observation 1. One could arbitrarily chose $\tau_{threshold}$.

$$\tau_{threshold} \to \infty, \exists e\ e \in \Sigma, \Sigma^* \to \Phi\ \ prob(e = false\_negative) = 1.0$$

and (2a)

$$\tau_{threshold} \to 0, \exists e\, e \in \Sigma^*, \Sigma \to \Phi\ prob(e = false\_positive) = 1.0 \quad (2b)$$

We chose $\tau_{threshold}$ as low as we can get away with so that the frequency of occurrence of false negatives is tolerable.

Let us assume the target system states to be a vector ϑ where

$$\vartheta = (\alpha_0, \alpha_1, \alpha_2 \ldots \alpha_i \ldots \alpha_k) \quad (3a)$$

where each α_i is an appropriate system variable that is expected to be affected when an intrusion event takes place either from within or without. We assume that for each variable α_i the measure α_i^n has a steady sampling distribution at an equilibrium state where it is observed and one can therefore obtain its higher order sampling statistics, if need be.

Lemma 1. At an equilibrium, $\forall i \forall n\ \frac{d}{dt}\alpha_i^n = 0$ $\to \alpha_i^n - E(\alpha_i^n) = T(0, \sigma_i^{(n)})$ where T(.,.) is some steady state distribution. The statistic $\sigma_i^{(n)}$ is the standard deviation associated with the sampling distribution statistics of the variable α_i^n.

The First-level NIDS filter design now emerges. Every first level NIDS filter is a conglomerate of a concurrent set of nk filters where k is the largest index value of the observable in the vector ϑ in (3a) and n is the largest order statistics the filter is equipped to handle. Note that this value n should not be too high lest the error in the computed statistics be too overwhelming, thus degrading the filter's efficiency.

We define an $IDS(i,p)$ filter as follows: The filter monitors the pth power of the variable α_i through the expression $\frac{abs(\alpha_i^p - E(\alpha_i^p))}{\sigma_i^{(p)}} \le k_i^p$. If for the current sample of the random variable $\frac{abs(\alpha_i^p - E(\alpha_i^p))}{\sigma_i^{(p)}} = m_i^p > k_i^p$, the *IDS(i,p)* filter issues an alarm to the observer with a strength $(m_i^p - k_i^p)/k_i^p$ indicating that a potential intrusion event is indicated at this variable with a belief index of $(m_i^p - k_i^p)/k_i^p$.

A comprehensive *IDS* filter may be articulated as

follows.

a. $IDS(i) : g_p((m_i^p - k_i^p)/k_i^p)$ where g_p is some suitable aggregating function like min_p, max_p, $total$

b. $IDS : f_i(IDS(i))$ where f_i is some appropriate aggregating function like g_p

3. ANOMALY DETECTION COPROCESSOR

The main impetus of the IDS filter is to respond to anomaly attacks when an anomaly is seen in a statistical sense primarily in terms of variance from a normal data flow from the network or outgoing to the network or percolating within a LAN. Anomaly is a straightforward event; it is anything that is outside the norm. In the discrete sample space spanned by packet header parameters, one may, for instance, encounter events e such that

$\forall header \exists param_1 \exists param_2 ... \exists param_i \; e = header(param_1, param_2, ... param_i) \in \Sigma^*$

Such events could be seen in some deterministic finite state automaton as an unaccepted language L [4]. There are several different types of commercial IDS system which can detect such strings *e* and flag them accordingly. Typically, one thinks of Snort, Emerald and Spade type packages [9] in this context, extending coverage using heuristics based on a number of rules. Using such systems concurrently at the same level as our IDS filter, we could reduce the chance of making false negatives (as in 2a). In terms of scope of the system at this coprocessor, we envisage, for example, detection of the following types of anomalous events. Obviously this is just an indication of what we want the coprocessor to cover.

1. For each TCP connection , check to see if
   a. 3-way handshake is properly followed. Flag the connection event e if
      $e \in$ {connection rejected, attempted: SYN-ACK never delivered, SYN-ACK received without initiating SYN}
   b. Data packets have corresponding ACK packets. Flag the event e if
      $e \in$ {unbalanced resent and ACK rates, hole rate, wrong data packet size rate, .. }
   c. Normal connection termination event. Flag the event e if
      $e \in$ {one or both sides do not send FIN, one uses RST, only one sends FIN}
   d. Proper TCP header. Flag the event e if
      $e \in$ {short TCP headers, port = 0 for either sender or receiver or both, (URG , PSH flag | no data packet), (SYN $\wedge$ URG), (SYN $\wedge$ PSH)}
2. At the IP level, flag the event e if
   a. $e \in$ {IP header < 20 octets, packet length < header length, (Low TTL value $\wedge$ $\neg$(limited broadcast)) …}
   b. $e \in$ {private IP addresses, "this address", "loopback address" | public internet}

4. CONCLUSION

Our paper outlines a low-CPU-usage, low-bufferbased network-, host- or router-centric intrusion detection system that is essentially an anomaly detector. This has the advantage that the limited buffer size would not become a bottleneck for the system. The basic architecture is iterative in that at each stage it would encounter a set of questionable events on its event list triggering an alert signal to the system administrator and posting the event to its Syslog file for further processing. To enhance its capability, it is suggested that a detection coprocessor be added to it so that the coprocessor mines out those significant questionable events which our IDS filter misses. The concept could be extended to include self-similar analysis to mine other indirect information.

5. REFERENCES

[1] Wenke Lee, Salvatore J. Stolfo, Philip K. Chan, Eleazar Eskin, Wei Fan, Matthew Miller, Shlomo Hershkop, Junxin Zhang. Real Time Data Mining-based Intrusion Detection. Proc. Second DARPA Information Survivability Conference and Exposition.

[2] Eleazar Eskin, Matthew Miller, Zhi-Da Zhong, George Yi, Wei-Ang Lee, Salvatore Stolfo. Adaptive Model Generation for Intrusion Detection Systems (2000). Proceedings of the ACMCCS Workshop on Intrusion Detection and Prevention, Athens, Greece 2000.

[3] Vern Paxon. Bro: A system for detecting network intruders in real-time. In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX 1998.

[4] K. Ilgun, R. A. Kemmerer, P. A. Porras. State transition analysis: A rule-based intrusion detection approach, IEEE Transactions on Software Engineering 21 (3) (March 1995). pp. 181-199.

[5] Sam Sengupta, Bruno Andriamanalimanana. Model abstractions for real-time network environment. Proceedings of SPIE, Vol. 4026, April 2000. pp. 212-220.

[6] Sam Sengupta, Bruno Andriamanalimanana. Domain-size constraint on real-time model abstractions. Proceedings of SPIE, Vol. 4367,

April 2001.

[7] Mohammed J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences, in Machine Learning Journal, special issue on Unsupervised Learning (Doug Fisher, ed.), Vol. 42 Nos. 1/2, Jan/Feb 2001. pp. 31-60.

[8] http://www.silicondefense.com/spice. SPICE is the product of Silicon Defense.

[9] Jack Koziol. Intrusion detection with Snort. SAM Publication.

***Sam Sengupta** is the professor of Computer Science Department at SUNY Institute of Technology, Utica, New York.*

His research activities and interests: Real-time network monitoring and management, Newtork protocols, Distributed Systems, Systems modelling, Operarting Systems. Current research activities include: Intrusion-detection techniques and security related issues; Fault-tolerance and survival using active networking; Mobile agent distributed architecture for large fast network.