International Journal of Research
e-ISSN: 2348-6848 & p-ISSN 2348-795X Vol-5, Special Issue-11
International Conference on Multi-Disciplinary Research - 2017 held in February, 2018 in Hyderabad, Telangana State, India organised by GLOBAL RESEARCH ACADEMY - Scientific & Industrial Research Organisation (Autonomous), Hyderabad.

Design & Development of Statistical Method - Exponentially Weighted Moving Average (EWMA) to Calculate Historical Averages for Factual Strategies and Learning Systems

Prof.Dr.G.Manoj Someswar[1], Dr.Venkata Reddy Medikonda[2]

1. Dean (Research), Global Research Academy – Scientific & Industrial Research Organisation [Autonomous], Hyderabad, Telangana State, India
2. Associate Professor, Department of CSE, Malla Reddy Group of Institutions, Hyderabad, Telangana State, India

Abstract

In this research paper, we give point by point depiction to the proposed systems and calculations utilized as a part of investigating behavioral differences. The general structure of BDA and usage of the analyzers are clarified too, to be joined by execution points of interest of different components in the proposed engineering.

This area is given to clarification of factual strategies and learning systems utilized by the analyzers, including observing of the difference estimations of the components utilizing exponentially weighted moving normal strategy, modifying those parameters that speak to reaction time difference of the servers, tuning measurable parameters, and performing incremental learning.

***Keywords:** Exponentially Weighted Moving Average (EWMA), Protocol Data Unit (PDU), Don't Fragment (DF), Time to Live (TTL), Distributed Denial of Service (DDoS)*

Introduction

We have used statistical method Exponentially Weighted Moving Average (EWMA) to calculate historical averages for feature values and differential values.

This strategy enables us to smooth out variances occurring in most system parameters. While in different inquires about spikes in parameters could be indications of assault, here we are for the most part intrigued by changes of differential parameters. The degree to which sudden changes are hosed is highlight subordinate.

Let x(p) speak to uncorrelated esteem perceptions for highlight Z, where p is the arrangement variable of the perceptions. Utilizing EWMA we ascertain the moving normal z(p)as

$$z(p) = \lambda\, x(p) + (1 - \lambda)\, z(p - 1),\ \ 0 < \lambda < 1$$

The average and standard deviation of z(p) are:

$$\mu_z = \mu_x$$

$$\sigma_z^2 = \sigma_x^2 \left(\frac{\lambda}{2 - \lambda}\right),$$

Where μ_x and σ_x are calculated during the training. Lower and upper control limits are

$$LCL_z = \mu_z - L\,\sigma_z, \qquad UCL_z = \mu_z + L\,\sigma_z$$

For a significance level of 5%, L = 1.96, and for 10%, L = 2.576. In detection (test), if z(p) falls outside [LCL, UCL] then the current average is far from the training average, and the case is considered to be an anomaly.

For features that count event-intensity over time unit, like packet rate, the research suggests transforming (1) into the following equivalent form:

$$z(t_j) = \lambda + (1 - \lambda)^{(t_j - t(j-1))}\, z(t_{j-1})$$

or example, if the feature is packets per second then for each event j, which is the arrival of a packet at time t_p, t_j is calculated as follows:

$$t_j = \text{floor}(t_p)$$

If, for instance, packet p_{j-1} arrives at 3.98s and packet p_j at 4.01s, then for $z(t_j)$, $t_j = 4$ and $t_{j-1} = 3$. In each series of packets, $z(t_{j-1})$ for the first packet comes from an experimental value.

It can be shown the same result at the end of every second.[1] The difference is that updates z(p) for every second, updates for every event (hundreds of events can take place within a second).

In case of fast and heavy-load attacks like DoS, sampling features every one second can open time gap for intrusive actions. For each feature Z to be monitored, two values z_{MIS} (p) and z_{SIS} (p) are calculated for every relevant Protocol Data Unit (PDU) p The di*ff*erential value, dz(p),

$$x(p) = diff(z_{MIS}(t_p - t), z_{SIS}(t_p))$$

Function di*ff*, in general, represents the comparison function for the behavior of two servers for feature Z. For a simple scalar feature such as packet length, this function is simply a subtraction. If the feature value is calculated using above formulae, then z(p) will replace $z(t_p)$ in formulae. The parameter t represents the delay between corresponding operations of MIS and SIS, and is discussed in detail in the next subsection.

Adaptive Response Time Difference

The parameter speaks to Adaptive Response Time Difference, and is under consistent change by application-level analyzers. This parameter systems level analyzers to consider the condition of servers at appropriate relative time, and assists application-level analyzers with matching answers from MIS and SIS. For each demand r that goes to MIS and is copied for SIS, application-level analyzers must have the capacity to coordinate the answers to r from MIS and SIS, to be named RMIS and RSIS .

In our organization that utilizations shadow customer for duplication, RSIS is generally observed after RMIS . [2]

The way toward coordinating, which is called Equalization, is a heuristic for generally applications. This is on account of BDA watches the conduct of servers autonomous of duplication component, so just server reactions are really being examined. Uncertainty to level RMIS and RSIS happens when the rate of comparable solicitations is high. For this situation, analyzers add their t parameters to recognizing components of PDU, and utilize them in the heuristics to determine identical reactions.

There is one non-recorded t parameter, which is utilized at system layers, and is being balanced by all application-level analyzers. In addition, every application-level analyzer keeps up its own Adaptive Response Time Difference parameter. For instance, tH is utilized just by HTTP Analyzer and balanced through HTTP PDUs.

Tuning Statistical Parameters

The analyzers should be calibrated before they are put into operation. Parameters and beginning qualities in EWMA recipes, introductory qualities for Adaptive Response Time Difference factors, resistance coefficients, and preparing interims for analyzers have been resolved. Every parameter is principally tuned in a way that makes the quantity of false positives least under no-assault runnings.

Resistance co-efficients are subject to the normal inertness of Shadow Client, and like starting qualities for H and Hx are accomplished tentatively through running tests with different informational collections. These parameters would be consequently adjusted to the states of specific organizations and systems in the early phases of test running. The way the consistent of EWMA equations is figured is clarified in the accompanying. [3]

It can be demonstrated that exponentially weighted moving normal channels are indistinguishable to first-arrange low-pass channels. Along these lines, we have utilized low-pass channels equations to decide the fitting an incentive for λ parameter in EWMA recipes. For each observed element Z, we run a (long) single preparing test with an underlying quality (e.g. $\lambda 0 = 0.1$) without any assaults in effect.

After the preparation time frame passes, we measure normal recurrence of false alerts for rest of the informational index, to be named fc. This parameter is identical to Turn Over Frequency of a low-pass channel. The time-consistent parameter of such a channel is ascertained as:

$$T_f = \frac{1}{2\pi \cdot f_c}$$

Then the λ parameter is

$$\lambda = T_f \frac{T_s}{+ T_s},$$

Where Ts speaks to unweighted normal of the interim between every estimation; i.e., landing interim of the PDUs, to be measured free of under-tuning parameters all through the trial. This parameter is a pointer of general traffic and in our investigations relies on upon the normal number of simultaneous clients (normal server stack).

Incremental Learning

Continuous changes in traffic example and normal load on servers are should have been reflected in the control estimations of the elements. In this manner, analyzers are required to experience incremental preparing and steady tests. When contrasted with established learning, in which input data is continuously used to extend the existing model's knowledge i.e. to further train the model. It represents a dynamic technique of supervised learning and unsupervised learning that can be applied when training data becomes available gradually over time or its size is out of system memory limits.[4] Algorithms that can facilitate incremental learning are known as incremental machine learning algorithms.

The aim of incremental learning is for the learning model to adapt to new data without forgetting its existing knowledge, it does not retrain the model.

Some incremental learners have built-in some parameter or assumption that controls the relevancy of old data, while others, called stable incremental machine learning algorithms, learn representations of the training data that are not even partially forgotten over time.

Steady changes in traffic example and normal load on servers are should have been reflected in the control estimations of the components. Accordingly, analyzers are required to experience incremental preparing and steady tests.

When contrasted with traditional learning, in which testing stage begins after that preparation stage is finished, here preparing never stops, as testing goes ahead with most recent parameters refreshed by the last stage (or interim) of preparing.

Incremental taking in is additionally different from k-overlay cross-approval as in the measure of test information is steadily expanding.

In k-crease cross-approval, the gave information is parceled into k subsamples, and each subsample is utilized once as the test information, while whatever is left of test is utilized as preparing information. The k results are then arrived at the midpoint of to deliver a solitary estimation.[5] In incremental picking up, moving normal is figured over chronicled preparing information, and test information is dependably the "current" examined information that will move toward becoming preparing information in the following interim.

Figure 1 delineates a case of incremental learning with preparing interims of 5 seconds. For each component, the preparation for control limits (LCL and UCL) of differential esteem is performed constantly as clarified in research paper, so is the trying of highlight differential esteem against those control limits. The arrangement of control breaking points (CL) is refreshed just toward the finish of every interim. At the end of the day, all tests in Testing i are performed against CL i, which has been taken

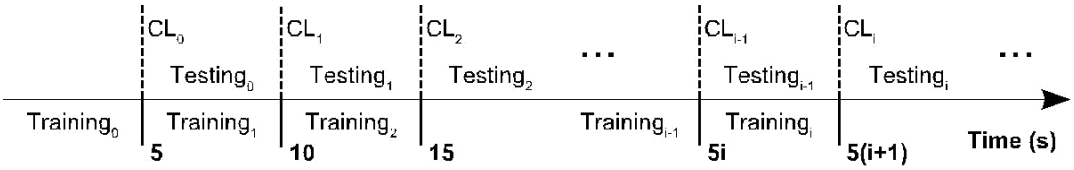

Figure 1: Incremental learning example

Implementation

In this area we clarify the execution of different components in the proposed engineering, including the shadow customers, the virtual customer, and the analyzers implanted in BDA. We sent a HTTP administration to complete trials and built up a calculation for the correlation of HTTP reactions.

Shadow Client

As specified before, the shadow customer screens Outer Segment to get those solicitations for which it has a duplicator customer. Application-level messages are created by reassembling lower-level bundles,

subordinate information for which we have utilized Ethereal (now is called Wireshark) apparatus [18].

Ethereal is additionally utilized for bundle catching and arrangement transformation. For each bolstered benefit, the relating convention customer makes another demand in light of the first demand for MIS, and sends it to SIS. In the event that appropriate, the customer must finish its communication with SIS; i.e., keep association alive until it gets the answer from SIS, and take additionally activities if the related convention requires such activities. At present, two customers are actualized: HTTP customer and ICMP Echo customer. Addendum C.2 demonstrates the UML outline and class depiction of the shadow customer execution.

HTTP Shadow Client

HTTP customer remakes any HTTP ask for that is sent to MIS, and afterward advances the reproduced demand to SIS. From the first HTTP ask for, shadow customer duplicates header fields to the shadow ask. Not all header fields are precisely duplicatable. For example, while a few fields like if-unmodified-since and acknowledge can be duplicated to the shadow ask for as they seem to be, others like host and if-match ought not be basically replicated without change, as they convey have

ICMP-Echo Shadow Client

ICMP Echo Request customer impersonates the first Ping solicitation and jam bundle length, Time to Live (TTL), and Don't Fragment (DF) highlights. This customer has not been enacted in the tests examined in lab.

Service Deployment

Three days worth of page get to log of the fundamental Web server at University of New Brunswick has been utilized as a part of our Web Service tests.[6] The got to pages have been put on MIS and SIS while the first chain of importance of registries were protected. Just freely accessible pages have been utilized as a part of the examinations.

Virtual Client

For the Web benefit, we built up a HTTP customer that peruses the Web server get to log and issues solicitations to MIS with a specific end goal to produce "ordinary" HTTP traffic for the servers. Figure 5.2 demonstrates a preview of the Web server get to log sections (Apache/2.0.54 on Solaris 9), in which the source locations are anonymized. Each log passage contains the accompanying data:

- Host name or IP address of the HTTP customer (anonymized in the figure).
- The time that the demand was gotten.
- The ask for line from the customer including HTTP strategy, asked for URL, and HTTP variant.
- Server status code; 200 means an effective reaction.
- Size of the question come back to the customer.
- The referrer URL.
- Identifying data that the customer program reports about itself.

The virtual customer utilizes just URL some portion of the above data. The planning amongst solicitations and the request of solicitations are controlled by parameters gone to the customer at startup. Taking after is the rundown of elements that our virtual customer offers:

- Virtually boundless number of simultaneous clients (HTTP customers) getting

to the Website.

- Time-interim between page solicitations can be settled, pseudo irregular, or random with Gaussian appropriation.
- URLs in get to log can be replayed consecutively or arbitrarily.
- Effect of load increment can be concentrated through determining the time interim between propelling new customers at startup.

In our tests, we instantiated 30 simultaneous HTTP customers getting to MIS haphazardly. Dispersion

of time-interim between page solicitations is Gaussian with μ = 3s and

σ = 900ms.

Index C.3 demonstrates the UML outline and class depiction of the virtual customer execution.

Overall Structure of BDA

Figure 3 demonstrates the general structure of BDA. Ethereal instrument [7] has been utilized for bundle catching and convention analyzation. The yield of dissectors is a grouping of Protocol Data Units (PDUs) that will be encouraged into PDU Preprocessor after arrangement transformation. PDU Preprocessor has two basic capacities:

1. Inspecting each PDU to see which layer and convention data is being conveyed by this PDU.
2. Pruning those parts of dismembered information that won't be utilized by the analyzers.

Differential Analysis Data Manager (DADM) gets the handled PDUs accompanied by convention data by means of Preprocessed PDUs line. Utilizing each PDU, DADM makes a question that satisfies necessities of differential analyzers. The information part of this protest is called Differential Analysis Data Unit (DADU), is essentially data about the fundamental and determined system elements, fundamental and inferred application highlights, regardless of whether this is an inbound or outbound parcel, is identified with MIS or SIS, conveys the data of which conventions, et cetera. [8]

DADM places DADUs into a buffer and sends the references to the analyzers. An analyzer will be informed of another DADU just if there is a few information there to be handled by that analyzer. For example, entry of an Echo Reply would trigger just MAC-level, IP, and ICMP analyzers.

International Journal of Research
e-ISSN: 2348-6848 & p-ISSN 2348-795X Vol-5, Special Issue-11
International Conference on Multi-Disciplinary Research - 2017 held in February, 2018 in Hyderabad, Telangana State, India organised by GLOBAL RESEARCH ACADEMY - Scientific & Industrial Research Organisation (Autonomous), Hyderabad.

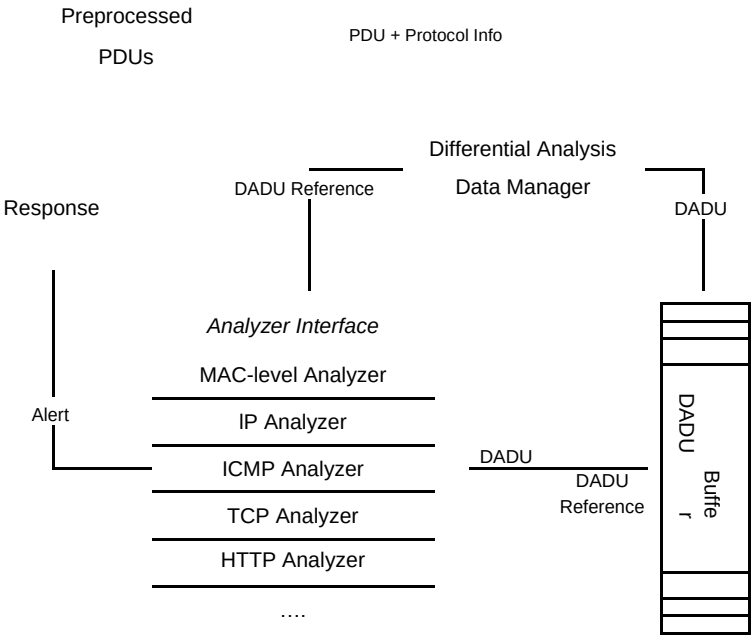

Figure 2: Overall structure of Behavioral Difference Analyzer

A DADU is rejected when every related analyzer have reviewed that information unit. Analyzers do their part for examination and issue endless supply of suspicious abberations amongst MIS and SIS exercises. Three analyzers are dynamic in investigations of Chapter 6: IP, TCP, and HTTP analyzers.

Index C.1 demonstrates the UML outline and class portrayal of the BDA implementation.

Analyzers

The analyzers that are actualized in BDA incorporate IP, TCP, ICMP, and HTTP Analyzers. Every analyzer screens its own particular feature(s) and furthermore may perform changes on the parameters that speak to reaction time difference.

IP Analyzer

The IP Analyzer monitors servers conduct at systems administration layer by observing proportion of traffic volume from server to customer over traffic volume from customer to server include. This element is named IP-TVR in compact. IP-TVR meets all criteria examined in Section. Traffic volume is measured as entirety of payload size of system bundles in one heading relating to one server.

Albeit some fundamental components, for example, IP bundle rate and IP parcel length appear to be great competitors at IP-level, our expanded tests demonstrated that they are touchy to retransmission of TCP lost or unacknowledged portions and burst operations of working frameworks, that outcomes in false positives. Such false alerts could be maintained a strategic distance from by drawing out the preparation time frame and tweaking the λ parameter in moving normal capacities, in any case, this would diminish the affectability of identifier in quick assault cases. [9]

The said highlight in IP Analyzer is foreseen to distinguish any assault that causes the proportion of traffic volume in MIS or SIS change, including many flooding assaults. Likewise when one of the servers has moved toward becoming host to malevolent code, e.g. a spying trojan, then we hope to see peculiarity in the differential estimation of this component in view of unpaired system exercises that the trojan starts.

This analyzer additionally checks if any IP correspondence happens amongst MIS and SIS. Provided that this is true, it issues and ready as in ordinary operation of the system the combined servers don't have to impart to each other.

ICMP Analyzer

The ICMP Analyzer explores answers to ICMP Echo (Ping) asks for, where ICMP answer rate is utilized as the observing component. Despite the fact that in genuine utilize, Ping solicitations are sent with different of seconds interims for system administration and investigating purposes, it may be abused for Distributed Denial of Service (DDoS) assaults or to misuse a defenselessness in convention stack (the illustration is Ping of Death adventure). The utilized element has demonstrated near none false encouraging points in ordinary use of Echo Requests, and effective discovery ability in ICMP flooding assaults, for example, Smurf.

Our tests demonstrate that different working frameworks have different strategies or capabilities in

taking care of vast ICMP parcels. For example, Mac OS X 10.3 on PowerPC equipment does not answer to ICMP Echo asks for that convey a payload bigger than 25,152 bytes. ICMP Echo asks for with enormous payloads are not basic in legitimate utilize and should be considered as suspicious exercises.

ICMP Analyzer does another imperative undertaking: ascertaining shadow dormancy for Ping-Pong exchanges and refreshing Adaptive Response Time Difference parameters (t and t ping). ICMP answers are coordinated by a key made out of goal IP address and bundle length. This analyzer has been deactivated in examinations clarified.

TCP Analyzer

The TCP Analyzer screens the level of exercises of servers at transport layer regarding the present number of dynamic associations and the rate with which servers finish their TCP associations effectively. We characterize two elements, named Active Connection Count and Connection Completion Rate to screen those main characteristics of servers. Our initial tests demonstrated that both create rather high rates of false positive; they are effectively impacted by blasted operations of the working framework as discussed about research paper.

We characterize a consolidated element named TCP Exertion as takes after:

$$\text{TCP Exertion} = \frac{\text{Connection Completion Rate}}{\text{Active Connection Count}}$$

This component figures the normal recurrence by which TCP executions be-gin and end their associations, and is a marker of TCP level of movement. It is autonomous of the transient server load, and shows low affectability to blast operations in opening and shutting of associations. In light of the definition, the feature would uncover assaults that cause inertia or intense exercises in transportation of parcels, and furthermore those that fumes assets of TCP layer.

The TCP Analyzer plays out an imperative checking too: if a demand for an association lands at a port not related with an open administration. In such cases, the analyzer issues a caution to show the recognition of an examining.

HTTP Analyzer

Like the ICMP Analyzer, the HTTP Analyzer checks conduct of the servers in regards to HTTP answer rate. Albeit heterogeneous servers may answer to incoming HTTP asks for in different orders, the answer rates for two servers are relied upon to be close at different times. The analyzer experiences periodical preparing to find out about the differential estimation of the answer rate include.

For the HTTP convention, the analyzer goes above and beyond and plays out a correct coordinating between answers from MIS and SIS. Since the sent site is the same on MIS and SIS in regards to both substance and structure, the HTTP Analyzer utilizes HTTP header data to locate the comparing answers to a demand. For every HTTP-Reply PDU, the analyzer manufactures a key from the accompanying things, if they are available:

- Status code
- Content length
- Content sort
- Requested URL (in the event of mistake)

The analyzer keeps up two hash tables, HM and HS , to monitor the answers every server conveys. These hash tables give mapping from the PDU keys to requested arrangements of PDU articles. For every HTTP-Reply PDU from the servers, the HTTP analyzer plays out the heuristic delineated in Figure.3.

The calculation empowers BDA to begin its examination whenever, even after the servers have begun their administrations. It additionally does not accept that for a demand, the answer from MIS will be dependably observed before the answer from SIS. On a server, there may be many administration

strings running and contending at the same time, and frequently the life cycle of each string is flighty. This implies despite the fact that a specific demand is continually landing at MIS before SIS, the answers may show up backward request.

On the off chance that for a specific key k gotten from a PDU (e.g. originating from SIS) there are as of now more than one coordinating PDUs from MIS, then that PDU will be selected and decided out that has the base time separation to the PDU from SIS, considering Adaptive Response Time Difference of HTTP:

$$t_s = \min_{t_n \in M} \{ | t_{now} - p - H | \},$$

where M is the set of all MIS PDUs match k, and t_s is the relative time stamp of the PDU to be eliminated.

In the event that the match of an answer is not seen before long, HTTP Analyzer will issue cautions. The resistance for a late match is dictated by two co-efficients, τ and T , T τ . The justification behind having two limits for issuing cautions is that notwithstanding when the postponement of a match goes past delay resistance (t • τ), it doesn't really show a specific glitch. A total coordinating answer may come at a later time that can be very late for any reason:

Moderate customer association and perusing from circle versus store in server are two regular reasons. Just redundancy of this occurrence in a tight cut of time demonstrates a noteworthy issue requiring consideration. The greater co-efficient, T , gives enough time to ensure that a specific answer never had a match. The reason not to sit tight from the earliest starting point for t • T to slip by is that having T as the sole resilience co-efficient would open a period crevice for quick meddlesome activities. [10]

International Journal of Research
e-ISSN: 2348-6848 & p-ISSN 2348-795X Vol-5, Special Issue-11
International Conference on Multi-Disciplinary Research - 2017 held in February, 2018 in Hyderabad, Telangana State, India organised by GLOBAL RESEARCH ACADEMY - Scientific & Industrial Research Organisation (Autonomous), Hyderabad.

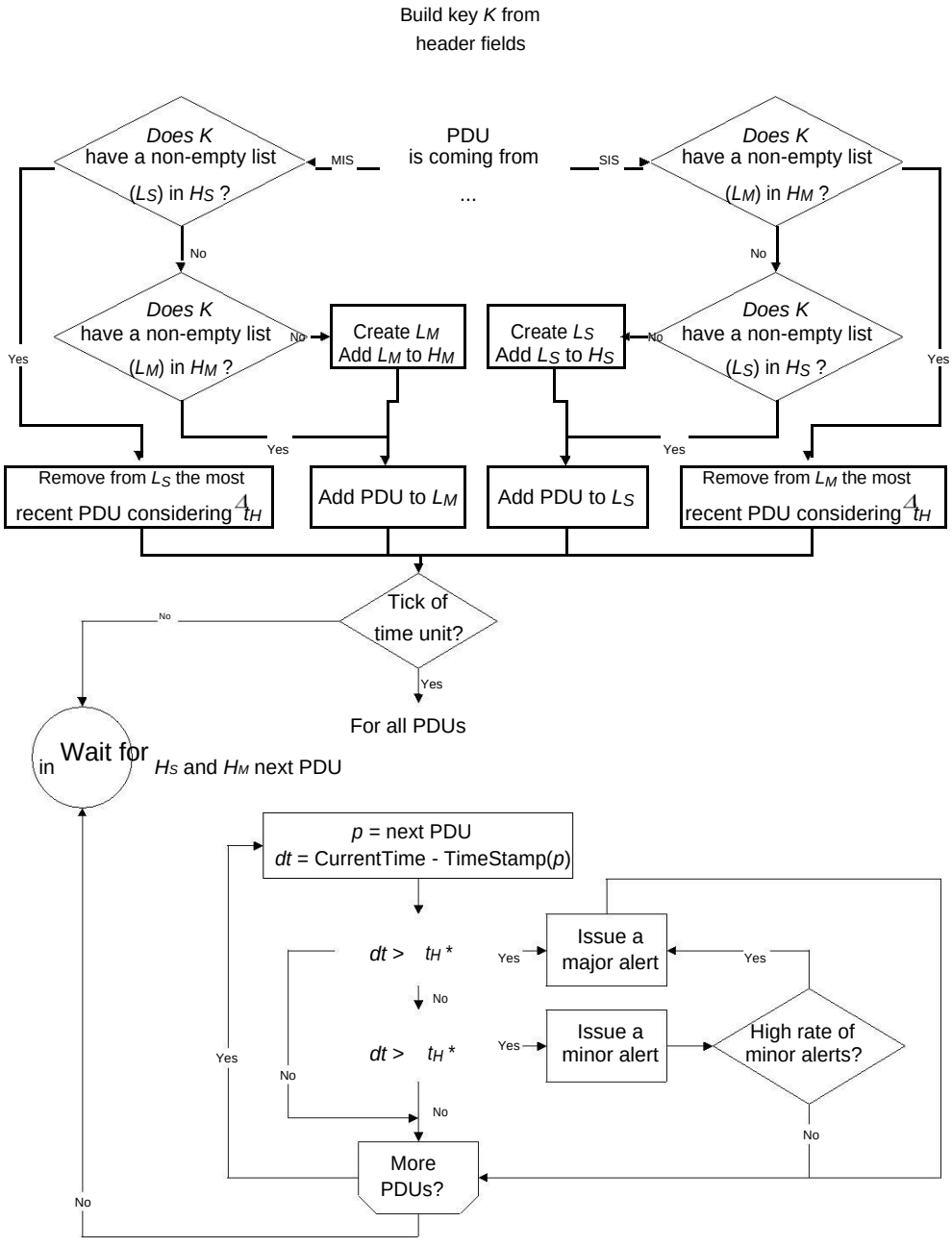

Figure 3: HTTP Reply Matching

In the event that the match of an answer is not seen before long, HTTP Analyzer will issue cautions.

The resistance for a late match is dictated by two co-efficients, τ and T , T τ . The justification behind having two limits for issuing cautions is that notwithstanding when the postponement of a match goes past delay resistance (t • τ), it doesn't really show a specific glitch. A total coordinating answer may come at a later time that can be very late for any reason:

Moderate customer association and perusing from circle versus store in server are two regular reasons. Just redundancy of this occurrence in a tight cut of time demonstrates a noteworthy issue requiring consideration. The greater co-efficient, T , gives enough time to ensure that a specific answer never had a match. The reason not to sit tight from the earliest starting point for t • T to slip by is that having T as the sole resilience co-efficient would open a period crevice for quick meddlesome activities. [10]

As specified some time recently, HTTP Analyzer additionally ascertains shadow idleness for sets of HTTP ask for answer and updates t and tH parameters. Just unambiguous demand answers will partake in this refresh; i.e., those PDUs that their keys guide to arrangements of size one.

Results & Conclusion

In this research work, we portrayed the calculations and factual strategies that are utilized as a part of the improvement of BDA. This incorporates observing of the element values utilizing EWMA, ceaseless change of versatile reaction time difference, tuning measurable parameters, and performing incremental learning in resulting stages. Execution subtle elements of BDA and different components of the proposed engineering are displayed too. We clarified how IP, TCP, ICMP, and HTTP analyzers capacity and how they add to the location procedure.

REFERENCES

1. Sankalp Singh, Michel Cukier, and William H. Sanders, Probabilistic validation of an intrusion-tolerant replication system, Proceedings of the Inter-national Conference on Dependable Systems and Networks (DSN ’03) (San Francisco, CA, USA), June 2003, pp. 615–624.

2. Ambareen Siraj, Rayford B. Vaughn, and Susan M. Bridges, Intrusion sensor data fusion in an intelligent intrusion detection system architecture, Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS’04) - Track 9 (Washington, DC, USA), IEEE Computer Society, 2004, p. 90279.3.(HICSS’04) - Track 9 (Washington, DC, USA), IEEE Computer Society, 2004, p. 90279.3.

3. S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C. L. Ho, K. N. Levitt, B.

Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, DIDS (Distributed Intrusion Detection System) - motivation, architecture and an early prototype, Proceedings of the 14th National Computer Security Conference (Washington, D.C.), October 1991, pp. 167–176.

4. Ming Tham, Dealing with measurement noise, http://lorien.ncl.ac.uk/ming/filter/ fillpass.htm, accessed December 2005.

5. Eric Totel, Frédéric Majorczyk, and Ludovic Mé, COTS diversity based in-trusion detection and application to web servers, Proceedings of the Recent Advances in Intrusion Detection Symposium (RAID'05), September 2005, pp. 43–62.

6.Thomas Toth and Christopher Kruegel, Evaluating the impact of automated intrusion response mechanisms, Proceedings of 18th Annual Computer Security Applications Conference (Las Vegas, NV, USA), IEEE Computer Society, December 2002, pp. 301–310.

7. Jeffrey Undercoffer, Anupam Joshi, and John Pinkston, Modeling computer attacks: An ontology for intrusion detection, Proceedings of the Sixth Inter-national Symposium on Recent Advances in Intrusion Detection (RAID'03), Springer-Verlag, LNCS 2820, September 2003, pp. 113–135.

8. Feiyi Wang, Fengmin Gong, Chandramouli Sargor, Katerina Goseva-Popstojanova, Kishor Trivedi, and Frank Jou, SITAR: A Scalable Intrusion-Tolerant Architecture for Distributed Services, Proceedings of the 2001 IEEE
Workshop on Information Assurance and Security (West Point, NY, USA),June 2001, pp. 38–45.

9. R. Wang, F. Wang, , and G. Byrd, Design and implementation of acceptance monitor for building scalable intrusion tolerant system, Software Practice and Experience 33 (2003), no. 1, 1399–1417.

10. Daniel Weber, A taxonomy of computer intrusions, Master's thesis, Massachusetts Institute Of Technology, Cambridge, MA, USA, June 1998.