Journal of Information Assurance and Security 3 (2008) 195-202

Anomalous Packet Detection using Partitioned Payload

Sandeep A. Thorat [1], Amit K. Khandelwal [2] , Bezawada Bruhadeshwar [3] and K. Kishore[4]

Centre for Security, Theory, and Algorithmic Research (CSTAR)
International Institute of Information Technology-Hyderabad, India
[1]*sandeep_thorat@students.iiit.ac.in*
[2]*khandelwal@students.iiit.ac.in*
[3]*bezawada@iiit.ac.in*
[4]*kkishore@.iiit.ac.in*

***Abstract*:** We present Anomalous Packet Detection using Partitioned Payload system, we call as AnPDPP. AnPDPP is an improvement to PAYL system which is considered one of the complete systems for payload based anomaly detection. PAYL takes into consideration the entire payload for profile calculation and effectively for anomaly detection. Payload length is very high on port numbers like 21 and 80. Hence it is difficult to apply PAYL on high speed, high bandwidth networks. We use CPP (Content based Payload Partitioning) technique which divides the payload into different partitions depending on content of payload. AnPDPP does payload based anomaly detection using a few CPP partitions. We demonstrate usefulness of the AnPDPP on the 1999 DARPA IDS data set. We observed 97.06% accuracy on port 80 using only 62.64% packet payload length with small false positive rate. This is a significant improvement over PAYL approach which uses 100% of the packet payload for anomaly detection.

***Keywords*:** Network Security, Anomaly Detection, 1-gram computation, packet payload, payload partitions**.**

1. Introduction

In recent era of information security systems all major network intrusion detection systems are still using signature based approaches for attack detection. Snort [1] and Bro [14] are popular example of signature based intrusion detection systems. Such systems use attack detection mechanisms based on signatures of already known attacks or vulnerabilities. This technique works well if the signature database is up to date. Signature based system are effective against the existing worms, viruses and network attacks. However, it has been observed that signature based detection mechanisms fail for zero day attacks or mutation of known attacks due to lack of signature availability when a zero day attack is launched. An alternative choice for this problem is anomaly detection systems [8]. An Anomaly-Based Intrusion Detection System, is a system for detecting computer intrusions and misuse by monitoring system activity and classifying it as either *normal* or *anomalous*. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out of normal system operation. In order to determine what attack traffic is, the system must be taught to recognize normal system activity. This can be accomplished in several ways, most often with artificial intelligence and machine learning techniques. Network based anomaly detection can be applied at different levels: protocol headers, packet payload. In AnPDPP we focus on packet payload based network anomaly detection. Such a system is useful for detecting payload based attacks like R2L, U2R, and worm infections. We found AnPDPP system useful for detecting suspicious packets arriving on the network which contain any payload based attack.

We propose a system which analyzes normal payloads for a particular service on a host and makes a set of payload profiles that are expected for that service. The payload's profiles are specific to the host and the communication behavior of the service; hence same profiles are not applicable across the different network environments. The system calculates byte frequency distribution using 1-gram based approach to make payload profile. The packet payload length has a strong impact on the byte frequency distribution, so multiple profiles are created for different payload lengths. Due to this, number of profiles for a particular service becomes very large. To minimize the complexity of profile comparisons, profiles are clustered together. The system is trained in an unsupervised way for profile creation. In the testing phase the system captures incoming payloads and compares the payload with stored normal profiles. If the new payload profile does not match with any stored profile for the same service, then an alert is generated indicating a suspicious packet. The system can be deployed at the network entry point level or at an end host. If we deploy the system for an individual host inside the network, the traffic profile of individual host is computed in the Training phase. This traffic profile may be significantly different than other host's traffic profile. Deployment of the system at network entry point makes sure that entire network's traffic profile is computed in the Training phase and we can apply anomaly detection mechanism for entire network. In our experiments we used DARPA 99 dataset in which data was collected at network entry point.

A similar approach is used by the PAYL [18] system and proved to be very useful for attack detection on port 21 and 80. But payload length observed on the port 21 and 80 is generally very high. And byte frequency computation for longer length payload becomes difficult on high speed networks. Also, if proper care is not taken in payload profiling such a system is vulnerable to mimicry attacks. Taking into consideration these facts, our system makes payload profiles depending on the content of the payload rather than using the same approach for all payloads. AnPDPP uses Content based Payload Partitioning (CPP) which was introduced and used by [12] and [17] for making partitions in the files according to the file content. CPP partitions the packet

Received August 20, 2008.

payload into a number of partitions and then the packet profile is computed on these partitions. We tested our system on the 1999 DARPA IDS data set [9] which is a commonly used data set by intrusion detection research community. In the 1999 DARPA IDS dataset the entire packet payloads are available which is useful for our system. We found 97.06% correctness in attack detection with 8.82% false positive rate in the results by profiling port 80 packets on 11 CPP partitions in payload. The 11 CPP partitions use 62.64% of payload length on average which is much less than payload length used by anomaly detection system PAYL [18] (which is 100%). This reduction in payload scanning length makes AnPDPP system operable in high speed networks. While partitioning CPP takes into consideration payload content, hence the system is robust against the mimicry attacks.

The rest of the paper is organized as follows. Section 2 discusses related work in network anomaly detection. In Section 3 we describe our system working in detail. Section 4 presents the experimental results and evaluations of the method applied to the 1999 DARPA IDS dataset. In Section 5 we conclude the paper and discuss future work directions.

2. Related Work

Presently in industry, rule based network intrusion detection systems such as Snort [1] and Bro [14] are most popular. These systems use signatures or finger prints to identify known attacks. But signature based systems are clueless in case of novel attacks where the attack pattern is not matching any stored signatures. Examples of such novel attacks are Zero day attacks, Mutation attacks etc. A Zero day attack is a computer threat that tries to exploit unknown computer application vulnerability. In attack mutation known instances of attacks are transformed into distinct instances which have same power of exploitation. Since attack signature is different than stored known signature due to transformations, such attacks are less likely to be detected by Signature based systems.

In such scenarios, anomaly detection systems differentiate between a 'normal' network activity and something other than 'normal'. These systems give better attack detection at the cost of high false positive rate. Network Anomaly detection systems such as PAYL [18], ALAD [10], PHAD [11], SPADE [3], NIDES [4] and NATE [16] compute statistical models for normal network traffic and generate alarms when the incoming traffic shows a large deviation from the normal model. These systems can be further classified depending on the features used to compute the normal models. Some of these systems viz. SPADE, NIDES and PHAD systems work with protocol headers. These systems use different features extracted from Ethernet, IP, ICMP, TCP, UDP packet headers for anomaly detection. These systems have shown better results for detecting protocol level attacks like scanning, probing etc. As these systems ignore the payload contents, they show very poor results for detecting application level attacks. Few network intrusion detection systems use connection and session level information for modeling profiles. Using connection and session level information for profile calculations makes profile's more accurate and false positive rate is decreased.

Some systems use few payload features for anomaly detection. NATE [16] uses first 48 bytes as a statistical feature starting from the IP header which includes at most the first 8 bytes of the payload of each network packet. But 48 bytes of payload is too less to detect any payload based attack. ALAD (Application Level Anomaly Detection) [10] uses features depending on the first word of each input line from the first 1000 lines of application payloads in addition to packet header features. ALAD is the first attempt to use "keywords" in the payload for anomaly detection, and it is used with other header fields to identify attacks. For any packet, the first word of each line will be extracted as a keyword. A packet could thus have multiple keywords. Several pairs of attributes are then created for modeling e.g. pairs like "source ip | destination ip" or "destination ip | destination port". For each pair, a statistical profile is constructed in the training phase. In the detection phase, packets containing new fields or keywords will get an anomaly score. The anomaly score is based on the assumption that if an experiment is performed 'n' times with 'r' different results, then the probability of the next new result is r/n. Once the anomaly score reaches the threshold, an alarm is generated. ALAD system tries to analyze the payload without prior knowledge just like our system, but it restricts profile to only the first 1000 lines.

Web based anomaly detection systems like [7] and [8] focus only on HTTP traffic and take advantage of known protocol format for constructing profile model. But these systems are not useful for detecting attacks on other protocols. Our system shares many characteristics with PAYL [18] which uses an approach based on payload byte distribution for profile computation. PAYL uses the entire payload for profile computation. Due to this, PAYL is not well suited for high speed networks with huge data traveling across the network. Our system aims to remove this weakness of the PAYL system by profiling payloads depending on CPP (Content Based Payload Partitioning).

3. Anomalous Packet Detection using Partitioned Payload

Anomaly detection systems run in two phases. In the training phase these systems profile normal behavior of network activity and store these profiles. In the testing phase such a system compares the current network activity profile with a stored profiles and reports alert when anything other than normal profile is seen on the network. For such a system following are major design goals [18]:

1. Generality of the system:
The system should be applicable for a broad range of applications or protocols. Due to this, anomaly detection systems which are protocol and service independent are always preferred.
2. Incremental Profiling:
Incremental profiling updates the computed profiles to accommodate changing communication patterns. The network activities keep on changing; the attack detection mechanism should be proactive to accommodate these changes.
3. Low false positive rate:

False positives are a major area of concern in anomaly detection systems. As these systems report alert for any thing other than the stored normal profiles; accuracy in detecting truly anomalous events is very important.

4. Resistance to Mimicry attacks:
In mimicry attacks, the attacker has access to the same information as the attack detection mechanism. Here the attacker attempts to know what is "normal" for the detection mechanism. Once this information is available, attacker crafts an attack to replicate the normal behavior. Such an attack may be considered as a normal packet payload by the anomaly detection system which is a serious vulnerability. Therefore the system should be resistant to mimicry attacks.

5. Efficiency to operate in the high bandwidth environments: As high speed, high bandwidth networks are becoming very common, anomaly detection mechanism need to be efficient enough to scan a huge amount of traffic.

6. Unsupervised Learning in the Training phase: Anomalous detection system requires the availability of labeled data in the training phase. Human errors are very much common in labeling a huge volume of data. These errors have very serious impact on correctness of the result. Due to this unsupervised learning is always preferred in any anomalous detection system for normal profile creations. Unsupervised learning requires very little or no human intervention in the training phase.

In past few years it has been observed that the above design goals are difficult to achieve together. Different systems attempt to balance these competing criteria's for payload anomaly detection. Our system adheres to incremental profiling, resistance to mimicry attacks, efficiency to operate in the high bandwidth environments and unsupervised learning.

3.1 AnPDPP System Architecture

The AnPDPP system architecture is shown in Figure 1. The system has two major components: Packet Profiler and AnPDPP Anomaly Detector. The Packet Profiler component deals with the normal packet profiling and creates the reduced profile model. The Packet Profiler module profiles packets in network environment or site specific way. So if we are interested to deploy system in different network; we need to carry out Packet profiling specific for that network. The Packet Profiler first partitions the payload of packet using CPP partitions and computes the packet profile using only initial few partitions. The Length-wise and Profile-wise clustering is applied to get reduced profile model. The reduced profile model is required for efficient comparison of the incoming network packet with stored profile. This reduced profile model is stored in database for applying Anomaly detection in Testing phase.

In AnPDPP Anomaly Detector module, incoming packet's profile is computed using initial few CPP partitions. This profile is compared with matching stored profile (having same destination port and similar payload length) and alerts are generated if the profile of the incoming packet is significantly different from stored profiles. The detailed working of each component is explained in Section 3.3 and 3.4. In AnPDPP, we model the payload using an n-gram analysis. An n-gram is a sequence of n adjacent bytes in a payload unit. A sliding window with width 'n' is passed over the whole payload and the occurrence of each n-gram is counted. Recently many security systems have started using n-gram analysis [2] for anomaly detection systems.

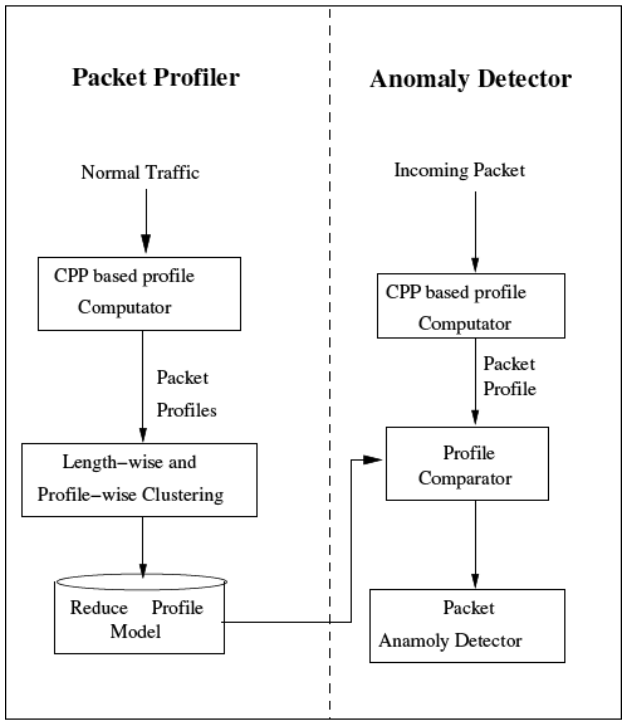

Figure 1. AnPDPP System Architecture

Our system creates the payload profiles using the 1-gram approach in an unsupervised way. A 1-gram model is considered as the simplest model for payload profiling. This simplicity is very useful while processing high voluminous network data. A 1-gram model requires a linear time scanning of the payload data and an update of a small 256 element histogram. This histogram stores byte frequencies observed for ASCII characters 0-255. PAYL has shown that the 1-gram model is sufficient and accurate for payload based attack detection.

As packet payloads show very wide variation in content, it is necessary to cluster packet payloads according to various criterions and then apply profiling on these clusters of packets. Our system does the payload clustering depending on the destination port number, length of the payload, direction of traffic (inbound or outbound), and byte frequency profile of the payload. This results in a much reduced profile model which is useful in the testing phase. Each network application has its own protocol and has its own payload type. This payload is site specific and varies with time so incremental profiling is required. Our system divides the packet payload into variable length content blocks using CPP (Content Based Partitioning). Before understanding the training phase for packet profiling we discuss CPP in next section.

3.2 Understanding Content Based Payload Partitioning

Content Based Payload Partitioning was introduced in the file system domain in LBFS [12]. Autograph [6] and

Earlybird [15] uses CPP for partitioning the payload at real time. CPP scheme determines the boundaries of each payload partition based on the payload content. It generates variable length content block from the payload. The generated partitions change a little under byte insertion or deletion from payload. This makes AnPDPP system robust against mimicry attacks.

CPP uses Rabin finger printing [13] to partition packet payload into content blocks. It computes a series of Rabin finger prints 'r_i' over a sliding n-byte window on the packet payload. It starts with first 'n' bytes in the payload and slides one byte at a time toward the end of the payload. It is efficient to find a Rabin finger print over a sliding window due to the linear complexity of computations. As CPP slides its window along the payload, it ends at a content block when 'r_i' matches a predetermined stopping criteria S. The average content block size produced by CPP is dependent on the user defined stopping criteria which is configured by the user. Autograph [6] and Earlybird [15] has shown that, we can choose the stopping criteria in different ways; it does not affect end result as long as we apply the same criteria in the training and the testing phase.

Rabin finger print function gives an integer value ('r_i') over a window size payload. If $r_i \bmod 1000$ is in the range of 550 to 600 (this is stopping criteria S we have chosen) we declare end of the current partition and move to next character to get another partition. Otherwise, we proceed to the next character after adding the present character to the current partition. As CPP decides content block boundaries probabilistically, CPP may generate very short content blocks. Very short content blocks do not represent byte frequency distribution of the payload properly causing payload profiling is most likely to be incorrect. Due to this in AnPDPP, we impose a minimum content block size limit and take care that this condition is satisfied by the CPP generated partitions. We gone through different stopping criteria's 'S' in our experimentations and we choose the stopping criteria where payload partitions are generated in sensibly uniform manner.

3.3 Training AnPDPP for Profile Creations

In the training phase, AnPDPP calculates the payload profile specific for a destination port. Since the payload length arriving at a particular port varies a lot, this causes variation in the payload profile. The different length payloads have different types of content; larger size payloads are more likely to have non-printable characters. Due to this, we compute a payload model for all different lengths for each port depending on direction of flow (inbound or outbound). For a real time traffic monitoring system, it is necessary to keep the profile model simple. We use the frequency of each ASCII character 0-255, which we call as 'byte frequency' for profile calculation. But some stable character frequencies and some variant character frequencies can result in the same average frequency. Hence, we compute the variance and standard deviation of each frequency as another characterizing feature. The average and standard deviation values of 256 characters from one profile are used to compare that profile with another profile. In the training phase we compute the profile for each specific length l of payload targeted to a destination port d. It computes $P_{d,l}$ model on packet payloads which is byte frequency of payload with length l targeted to port d. This generates huge number of profiles as for every small length variations we are computing profiles differently. To reduce information stored in the profile model we apply different clustering techniques on these profiles. These clustering techniques are length-wise clustering and profile-wise clustering. AnPDPP initially applies length wise clustering, which combines profiles of length l_i and l_j together where difference between the lengths l_i and l_j is less than threshold l_t. The resultant profile has byte frequency distribution and length equal to the average payload length of combined profiles. The value of l_t is kept user configurable. Once length-wise clustering is done profile-wise clustering is applied. The profile-wise clustering pays attention on sparse profiles. In sparse profiles, very few numbers of packets participate in the calculation of the profile. In profile-wise clustering, sparse profiles are merged with their nearest matching profile. The nearest matching profile is found by taking into consideration the Manhattan distance m_d of the profiles from each other. If the m_d is less than threshold m_t, then the two profiles are combined together using the same technique used in length-wise clustering. The value of m_t is user configurable. The graph in Figure 2.A and Figure 2.B shows normalized average byte frequencies observed on port 80 and port 21 for characters 0-255 before applying CPP.

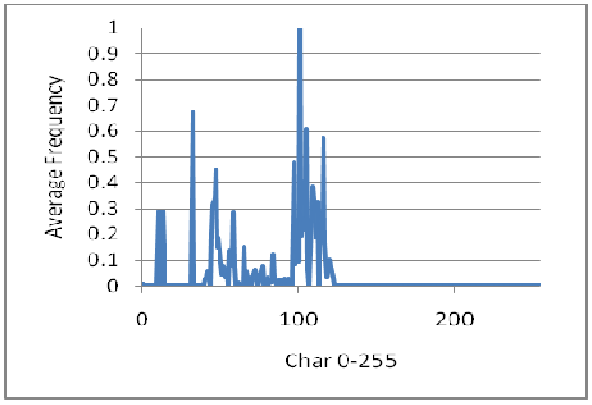

Figure 2.A. Normalized average byte frequency distribution on port 80 using entire payload

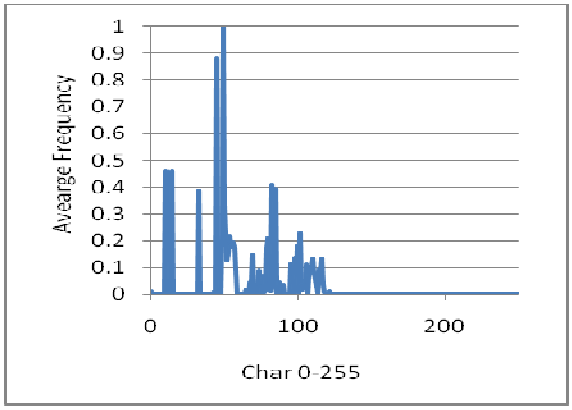

Figure 2.B. Normalized average byte frequency distribution on port 21 using entire payload

Up to this point we use similar techniques for payload profiling with PAYL. But PAYL uses the entire payload for profile computations, which is difficult to apply in high speed, high bandwidth networks. Our system computes profile depending on only the initial N partitions we got after applying CPP on payload. The value of N is user configurable. As described in the above section, CPP does payload partitioning depending on the content of the payload in protocol independent way. CPP gives an offset in payload which is the last position of the N^{th} partition. We compute the payload profile by using payload length up to this offset. We kept the number of partitions to be used 'N', configurable in system and tested system for correctness of the results. Figure 3.A shows the byte frequency graph by taking into consideration 11 partitions (we got most correct results for 11 number of partitions) of payload on port 80. As we can see graphs in Figure 2.A and Figure 3.A similar nature and we are not losing any profile relevant information after using 11 CPP partitions. We can also see similar nature of graphs in Figure 2.B and figure 3.B which represents results on Port 21.

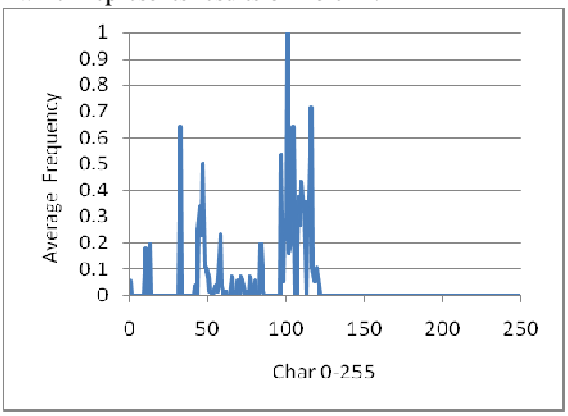

Figure 3.A. Normalized average byte frequency distribution after applying CPP on port 80 using 11 partitions in CPP

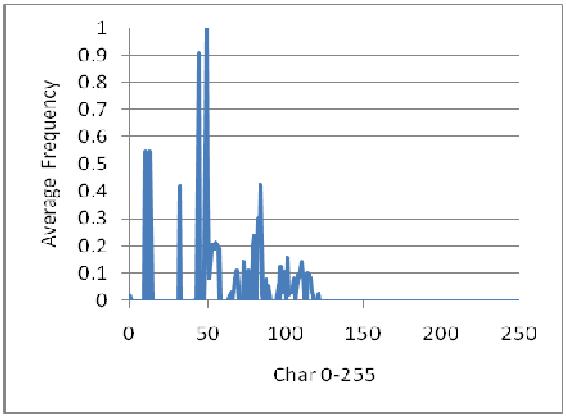

Figure 3.B. Normalized average byte frequency Distribution after applying CPP on port 21 using 11 partitions in CPP

3.4 Testing AnPDPP for Attack Detections

In the testing phase, AnPDPP compares the incoming packet payload profile with the stored profile $P_{d,l}$ associated with same destination port and having a similar payload length. This comparison is done by finding out the simplified Mahalanobis distance between the stored profile and the new profile. If incoming payload is not matching with any of the stored profile, then an alert is generated notifying a suspicious packet arrival. Here two possibilities are present: If incoming Payload has entirely different payload length than stored profiles then alert is generated. Otherwise if incoming payload has length similar to one of stored profile and Mahalanobis distance between two profiles is more than lengthwise threshold value l_t an alert is generated. In the testing phase also we use CPP to partition the incoming payload into N partitions. AnPDPP takes into consideration payload length up to the N^{th} partition to compute the profile of the payload. Here, the value of N is equal to the one used in the training phase.

To find out the distance between the stored profile and the incoming packet payload profile, we use the simplified Mahalanobis distance which is given by PAYL system [18]. Mahalanobis distance is a standard distance metric to compare two statistical distributions. It is a very useful way to measure the similarity between the new payload sample and the previously computed model. Here we compute the distance between the byte distributions of the newly observed payload against the profile from the model computed for the corresponding length range. If the distance is higher then it's more likely this payload is abnormal. Mahalanobis Distance is given by following formula:

$$d^2(A,\overline{B}) = (A-\overline{B})^T C^{-1}(A-\overline{B})$$

where B and $\overline{A}$ are two feature vectors, vector B is the feature vector of the arriving payload, and $\overline{A}$ is the feature vector computed in the training phase. And C^{-1} is inverse covariance matrix $C_{i,j} = \mathrm{cov}(B_i, B_j)$ B_i and B_j are i^{th} and j^{th} element of training vector. Each profile is considered as a feature vector of 256 (ASCII values) elements. The advantage of Mahalanobis distance is that, it

takes into account not only the average values but also variance and the covariance of the variables. The bytes of network packet are statistically independent. Thus, the covariance matrix C becomes diagonal and the elements along the diagonal are just the variance of each byte. In computing the Mahalanobis distance high price is paid to compute multiplications and square roots after summing the differences across the byte value frequencies. After noting down these things PAYL has given simplified Mahalanobis distance formula as:

$$d(B,\overline{A}) = \sum_{i=0}^{255} (\left|B_i - \overline{A_i}\right|) / (\sigma_i + \alpha))$$

σ_i is standard deviation on i^{th} ASCII characters while α is smoothing factor added to remove possibility of distance becoming infinity when σ_i is zero.

In the testing phase, AnPDPP does incremental profiling as well. If the arriving payload was found to be normal with comparison to some stored profile, then that stored profile is combined with arriving payload profile using technique used in the length-wise clustering.

4. Experimental Setup and Results

We tested AnPDPP on the 1999 DARPA IDS data set [9] which is considered as a standard data set to evaluate intrusion detection systems. The DARPA data set is collected by MIT Lincoln Labs for evaluation of Intrusion Detection Systems. This data set has been used for many research works on network Intrusion Detection Systems and network systems. The data set consists of three weeks of training data and two weeks of test data. Although there are problems due to the nature of the simulation environment that created the data, it still remains a useful set of data to compare techniques [5].

In our experimental setup, we initially implemented payload based anomaly detection system which profiles the packet taking into consideration the entire payload. We conducted experiments using each packet as the data unit which were inserted in MYSQL database. For this we implemented 'extractor' module which inserts DARPA data set into MYSQL database after reconstructing TCP connections from network Packets. Since most of Internet application uses TCP protocol and for reduction in complexity of experiments we focused our experiments only on TCP traffic for anomaly detection. We examined only the inbound TCP traffic to the ports 0-1023 of the hosts 172.16.xxx.xxx. We implemented our front end application program in C++ which carries out Training and Testing of system using Packets available in MYSQL database.

	Payload Profiling Using Full Payload	**Payload Profiling After Applying CPP**
Number of Attacks Detected	61	60
Average % of Payload Considered	100	61.13
Average % of False Positive Rate	0.0623	0.1407
Time required to process 100 Packets in Training Phase	102.592ms	101.269ms
Time required to process 100 Packets in Testing Phase	0.5171ms	0.5014ms

Table 1: Comparison of results Before and After CPP Partitions for Payload Profiling

We trained the system on the DARPA dataset using data of week 1 and week 3 and then evaluated the detector on data of weeks 4 and 5. We carried out experiments using full payload as well as using CPP. As shown in Table 1, we compared both the methods in terms of number of attacks detected percentage of payload use, average false positive rate and time taken to process 100 packets during training as well as testing phase. It is found that in case of full payload profiling the number of attack detected is 61 out of 62 after using 100% of payload on port 21 and port 80. Thus using full payload 98.39% of attacks was detected. In contrast, the CPP uses only 61.13% of the payload and detects 60 attacks out of 62. The average false positive rate, in case of full payload is 0.0623%. In case of CPP, it increases to 0.1407%. The time to process 100 packets in training phase is 102.592ms and 101.269ms in case of full payload and CPP respectively. In testing phase the time taken to process 100 packets is 0.5171ms and 0.5014ms respectively for full payload and CPP. Thus we observe a slight decrease in the processing time both during the training and testing phase. The Training and Testing phase included additional component for partitioning the payload. In this component we need to find out a sequence of Rabin Finger prints over the Payload. Though Rabin Finger print calculation is linear computation, extra time is spent in it. Hence decrease in time observed for Training and Testing phase is comparatively less as compared to PAYL system. We kept number of CPP partitions to be taken into consideration for payload profiling configurable. We got best result at 0.001 smoothing factor. The rest of configurations like l_t, m_t smoothing factor etc., are kept exactly similar as that of previous system (where whole payload is taken into account for profiling) implementation and observed correctness of result for various choices of N. The values of threshold variables like l_t, m_t, α are very much sensitive to training data.

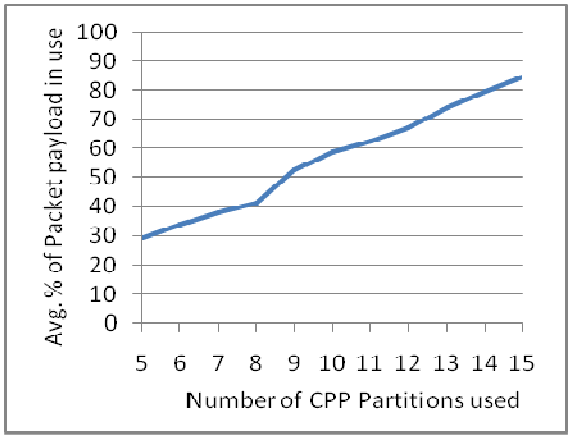

Figure 4. Percentage of payload used goes on increasing as we increment number of CPP partitions on port 80.

Figure 4 shows, as we increase number of partitions, the average payload size taken into consideration for profile computation goes on increasing. Also, we observed an increase in the correctness of results with increase in number of CPP partitions taken into consideration for payload profiling. Figure 5 shows the relevant results.

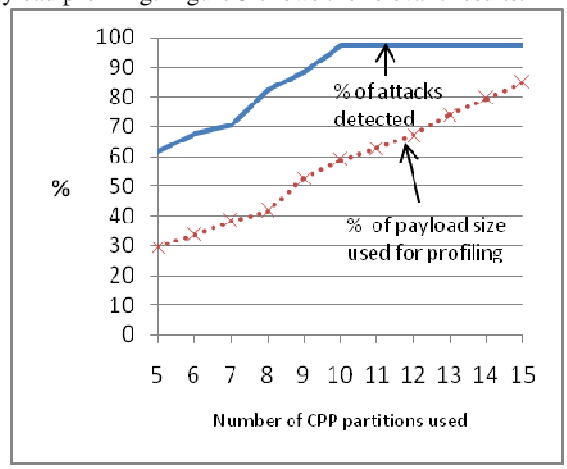

Figure 5. Percentage of attacks detected and % of payload size used by varying CPP partitions on port 80.

For 11 partitions, we found best results with 97.06% of attacks detected correctly and using an average 62.64% of packet payload. False positive rate observed for these configurations is 0.17%. This indicates that we could save 37.36% of packet processing on port 80 which is a significant improvement over PAYL. We got similar success on port 21, where 96.43% of attack detected using average 60.12% of packet payload used and a false positive rate of 0.1114% when 11 partitions are used.

5. Conclusion

In this paper, we proposed a system which can detect anomalous packet payloads without taking into consideration entire payload length. This is an improvement over systems like PAYL which considers whole payload for anomaly detection. The experimental results prove our method is good in attack detection on port 21 and port 80. As AnPDPP uses Content based Payload Partitioning (CPP), the system is safe against mimicry attacks and Mutation attacks. Also AnPDPP do not require any preprocessing on header of incoming packet which reduces time.

Presently our system shows poor results at ports like 22, 23, and 25. Also, false positive rate is little higher after using CPP partitions for profiling which may be due to random nature of data at these ports. Similar limitations are shown by other Payload based network anomaly detection systems. If we combine our technique with header level, session, and connection based information then false positive rate can be brought down. As future work we plan to revise the partitioning criteria in port specific manner in order to apply this technique successfully on other ports like 22, 23, and 25. Also in future we plan to apply payload partitioning technique for Anomaly Detection for UDP protocol.

Acknowledgment

We would like to thank all faculty members and research students of CSTAR research laboratory in IIIT Hyderabad. We are very much thankful to Prof. V Ch Venkaiah and Prof. Kannan Srinathan for their overall support and helpful suggestions throughout the work. We are also very much thankful to Anuj, Amol, Rohit for useful discussions on the project work.

References

[1] Snort: The open source network intrusion detection system

[2] M. Damashek. “Gauging similarity with n-grams: language independent categorization of text”. *In Science,* 267(5199), pages 843–848.

[3] J.Hoagland. Spade. In Silican Defense, http://www.silicondefense.com/software/spice, 2000.

[4] H.S.Javits and A.Valdes. “The nides statistical component: Description and justification”. *In Technical report, SRI International,* Computer Science Laboratory, 1993.

[5] Salvatore J. Stolfo Ke Wang, Gabriela Cretu. “Anomalous payload based worm detection and signature generation”. *In Proceedings of the Eighth International Symposium on Recent Advances in Intrusion Detection RAID* 2005,pages 227– 246, 2005.

[6] Hyang-Ah Kim and Kart B. “Autograph: Toward automated, distributed worm signature detection”. *In Proceedings of the 13th Usenix Security Symposium*, pages 271–286, 2004.

[7] Christopher Krgel, Thomas Toth, and Engin Kirda. “Service specific anomaly detection for network intrusion detection”. *In Proceedings of the 2002 ACM symposium on Applied computing*, pages 201–208, 2002.

[8] C. Kruegel and G.Vigna. “Anomaly detection of web-based attacks”. *In Proceedings of 10th ACM Conference on Computer and communications Security CCS’03*, pages 251–261, October 2003.

[9] R. Lippmann. “The 1999 DARPA offline intrusion detection evaluation”. *In Computer Networks*,34(4), pages 579–595, 2000.

[10] M. Mahoney. "Network traffic anomaly detection based on packet bytes". *In Proc. ACM-SAC, Melbourne FL*,pages 346– 350, 2003.

[11] M. Mahoney and P. Chan. "Learning non stationary models of normal network traffic for detecting novel attacks". *In Proc. SIGKDD 2002,* pages 376–385, 2002.

[12] Athicha Muthitacharoen, Benjie Chen, and David Mazieres. "A low bandwidth network file system". *In Symposium on Operating Systems Principles*, pages 174–187, 2001.

[13] Rabin M.O. "Finger printing by random polynomials". *In Tech. Rep. TR-15-81, Center for Research in Computing Technology, Harvard University*, 1981.

[14] VernPaxson. "Bro: a system for detecting network intruders in real-time". Computer Networks (Amsterdam, Netherlands: 1999), 31(23-24):2435–2463, 1999.

[15] Singh S., Estan C.,Varghese G., and Savage S. "The earlybird system for real-time detection of unknown worms". *In Tech. Rep.CS2003-0761, UCSD*, 2003.

[16] C.Taylor and J. Alves-Foss. "An empirical analysis of nate: Network analysis of anomalous traffic events". *In 10th New Security Paradigms Workshop*, 2002.

[17] U.Manber. "Finding similar files in a large file system". *In Proceedings of the USENIX Winter Technical Conference*, pages 1–10, 1994.

[18] K.Wang and S. Stolfo. "Anomalous payload-based network intrusion detection". *In Recent Advances in Intrusion Detection, RAID*, pages 203–222, September 2004.

[19] TCPDUMP and LIBPCAP Project: http://www.tcpdump.org/

[20] MIT Lincoln Lab DARPA website : http://www.ll.mit.edu/IST/ideval/data/data_index.html

Author Biographies

Sandeep A Thorat completed his BE from Shivaji University Kolhapur. He completed MTech from IIIT Hyderabad with specialization in Information Security. His area of interest includes Network and Information Security. Presently he is working as faculty in Computer Science Department of RIT Engineering College.

Amit K. Khandelwal completed his B.E . from Government Engineering College, Kota, India and M.Tech. from IIIT Hyderabad, India with specialization in Information Security. His area of interest are data mining, network intrusion detection system and Buffer overflow attacks. Currently he is working in nVIDIA Graphics Pvt. Ltd. India as System Software Engineer.

Bezawada Bruhadehwar is working as Assistant Professor in IIIT Hyderabad. He completed his PhD from Michigan State University in 2005. His Area of interest includes Network Security, Key Management, Secure Group Communication, Adhoc Networks, and Secure Storage Systems.

Kishore Kothapalli is working as Assistant Professor in IIIT Hyderabad. He completed his PhD from John Hopkins University. His Area of interest includes Theory of Network Communication, Overlay Networks and Distributed Algorithm.