**Network Traffic Anomaly Detection in Embedded Systems**

1 Introduction

Network traffic analysis is the process of recording, reviewing and analyzing network traffic for the purpose of performance, security and/or general network operations and management [1]. Anomaly detection in network traffic can be used for identifying network problems, changes in network activity or intrusion attempts.

Network traffic anomaly detection in embedded systems is challenging because it must be accomplished in real time using limited resources, such as memory, storage and processing.

In this paper we propose a technique for performing network traffic anomaly detection in embedded systems that examines packet header fields and identifies anomalous packets as those with field values that haven’t been observed for a long period of time.

2 Background and Definitions

Intrusion detection is one use of network traffic analysis for network security. Intrusion detection can be accomplished via signature-based or non-signature-based approaches. Most intrusion detection systems are signature-based, which means that they search for a known identity, or *signature*, for each specific intrusion event, and they depend on receiving regular signature updates [2]. Non-signature-based intrusion detection systems are also known as anomaly-based intrusion detection systems. According to Wikipedia, “an *anomaly-based intrusion detection system* is an intrusion detection system for detecting both network and 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 attempts to detect any type of misuse that falls out of normal system operation. This is as opposed to signature-based systems, which can only detect attacks for which a signature has previously been created” [3].

A number of open source intrusion detection systems (IDS) are available [4]. One of the most popular of these is Snort, which was originally developed by Martin Roesch [5,6]. It is a signature-based IDS that uses a rule-based traffic collection engine. A number of extensions and plug-ins have been written for Snort, some of which provide non-rule-based anomaly detection, such as Spade [7].

Much academic research has been done on anomaly detection in network traffic. Some network anomaly detection algorithms use artificial intelligence techniques, such as machine learning or neural networks [8,9]. Others use statistical analysis techniques such as forecasting, principal component analysis and smoothing [10,11,14]. Most anomaly detection algorithms use training data to set parameter values or probabilities that are used by the algorithm.

A number of researchers use packet header data, such as IP addresses and port numbers, to establish patterns of normal and anomalous behavior in network traffic [7,12,13,14]. The advantage of using packet header data for anomaly detection in embedded systems is that the size of the data is small.

Several methods, including the ones in [7,12,13] use frequency-based and time-based models. In a *frequency-based model*, the probability that an event will happen is determined by the frequency that it occurs during training. In a *time-based model*, the probability of an event depends on the time since it last occurred. When an event occurs after the training period, its anomaly score is inversely proportional to its probability. If the anomaly score is greater than a given threshold, then the event is reported as anomalous. The definition of an event differs depending on the method. In some methods, an event is the value of a particular combination of header fields, such as (destination address, destination port). In [13] an event is the value of a particular byte of the header.

3 Requirements

We define the following requirements for our network traffic anomaly detection process in an embedded system:

* It must operate in real time and process data packets as they pass through the system, rather than storing data and analyzing it offline.
* It must use a small amount of data storage.
* It must model normal versus anomalous network data packets using information in the packet header fields.
* It must be able to revise its model for “normal” data packets if network traffic patterns change.

4 Model and Assumptions

We assume that we have access to bi-directional network traffic over a wire on a network. We assume that the processing of layers 1 and 2 in the Open System Interconnection (OSI) model [15] has been done and we are given as input data packets starting at the network layer (layer 3) in the OSI model.

We use a model that is both frequency-based and time-based, similar to the ones used in [12,13]. We model a data packet using the values in its header fields in layers 3 and 4 in the OSI model (TCP/IP, UDP/IP, etc. header fields). The checksum fields aren’t used. We use the values of individual fields, as well as the following combinations of fields: (source address, destination address) for IP packets and (source address, source port), (destination address, destination port), and (source port, destination port) for TCP/IP and UDP/IP packets.

During the initial training period, summary data is compiled, but no packets are identified as anomalous. For each field or field combination, we store the last time the field was seen, where time is represented as the number of packets processed. We also store the total number of times that the field/field combination was seen.

At the end of training, the frequency of a particular field/field combination is *n/p*, where *n* is the total number of times that the field/field combination was seen and *p* is the total number of packets processed. In a frequency-based model this value can be viewed as the probability that a particular field will appear in a packet after the training period. In a time-based model the probability that a particular field will appear in a packet after the training period is *1/(p-q)*, where *p* is the total number of packets processed during training, and *q* is the last packet number where the field was observed. In other words, the longer the time since the field appeared in a packet, the smaller the probability that it will appear in next packet observed.

We perform similar calculations for field and field combination values. For each field or field combination value that is observed during training, we store the last time (i.e. packet number) that the value was seen and the total number of times that the value was seen. At the end of training, the frequency that a particular value appeared in a field is *m/n*, where *m* is the total number of times that the value was seen and *n* is the total number of times that the field was seen. In a frequency-based model this value can be viewed as the probability that a particular value will appear in a field after the training period. In a time-based model the probability that a particular value will appear in a packet after the training period is *1/(p-q)*, where *p* is the total number of packets processed during training, and *q* is the last packet number where the value was observed.

After the initial training period, we use a combined frequency-based and time-based model for computing the anomaly score of a packet. For each field or field combination value in the packet, we compute a frequency score that is equal to the inverse of the frequency probability; that is, *n/m*, or *p* if the value was never seen before. The frequency score will be in the range [1..*p*]. We also compute a time score that is equal to *p-q*, or *p* if the value was never seen before. The time score will be in the range [1..*p*]. The anomaly score for a field/field combination value is the product of its frequency score and time score, which is in the range [1..*p2*], where a higher anomaly score indicates a more anomalous field value. We normalize the anomaly score to a value in the range (0,1] by dividing it by *p*2. The anomaly score for the packet is the maximum anomaly score for all field values in the packet. We use maximum rather than average because we want to be able to detect packets where only a single field value indicates an anomaly. If the normalized anomaly score for the packet is greater than a given threshold, then the packet is reported as anomalous.

5 Algorithm for Training Period

In this section we describe the basic algorithm for the training period.

Tables F and V are used to store field and data summary values for the training period. For table F, each field and field combination that is used in the model is assigned an index in the table. For example, IP version field = 1, IP IHL field = 2, IP type of service = 3, etc. Since there are a limited number of packet header fields, this table can be of fixed size. The value for F[i] is (m, n), where m is the number of the last data packet that contained this field and n is the total number of occurrences of this field in the data packets. The table F is initialized to have F[i] = (0, 0) for all i.

Note that a particular field or field combination may occur multiple times in a data packet if IP tunneling is used. In this case, F[i][n] will be incremented for each occurrence of a field. However, only field combinations within a single header will be counted. For example, if there are two IP headers in a data packet, F[i][n] will be incremented for (source address, destination address) in each IP header, but we won’t increment F[i][n] for a source and destination address combination from two different IP headers.

A hash table is used for V since we don’t want to store all possible field values. For each field or field combination stored in F, a hash key k for V is created from the index i used in table F and the field value found in the data. The value for V[k] is (q, r), where q is the number of the last data packet that contained this field value, r is the total number of data packets that contained this field value.

For each data packet d do

Increment p, the total number of training packets processed so far.

For each field and field combination i that exists in d and F do

F[i][m] = p

F[i][n]++

Compute the hash key k for i and the field value

If V[k] doesn’t exist, V[k] = (p, 1)

Else if V[k] does exist

V[k][q] = p

V[k][r]++

6 Algorithm for Anomaly Detection Period

During the anomaly detection period we use the tables F and V to compute an anomaly score for each new data packet. If the score is greater than a given threshold T, then it is reported as an anomaly. The threshold T is a number in the range (0,1). Ideally we would like to select a threshold that only reports true anomalies and doesn’t miss detecting any anomalies. While it’s probably impossible to select a threshold that yields completely correct results, the value of T can be adjusted after analyzing initial data results from the algorithm to produce the best possible results. The lower the value of T, the larger the number of false positives produced by the algorithm, and the higher the value of T, the larger the number of false negatives produced by the algorithm.

Also during the anomaly detection period the tables F and V are updated with the information in the data packet.

For each data packet d do

Increment p, the total number of packets processed so far.

For each field and field combination i that exists in d and F do

Compute the hash key k for i and the field value

Compute the normalized anomaly score A for the field value as follows

time\_score = p - V[k][q] if V[k] exists, otherwise p

frequency\_score = F[i][n] / V[k][r] if V[k] exists, otherwise p

anomaly\_score = time\_score \* frequency\_score

A = normal\_score = anomaly\_score / p2

If A > T, then mark the data packet as anomalous and add the field and value to the anomaly report for the data packet.

Update the F and V tables as was done during training

If the data packet was marked as anomalous, then report it. The report contains the entire contents of the data packet with the anomalous field values marked.

7 References

1. <https://www.techopedia.com/definition/29976/network-traffic-analysis>
2. Brox, Arnt, “Signature-Based or Anomaly-Based Intrusion Detection: The Practice and Pitfalls”, SC Magazine, May 2002 <https://www.scmagazine.com/signature-based-or-anomaly-based-intrusion-detection-the-practice-and-pitfalls/article/548733>
3. <https://en.wikipedia.org/wiki/Anomaly-based_intrusion_detection_system>
4. Schreiber, Joe, “Open Source Intrusion Detection System Tools: A Quick Overview”, Alien Vault, Inc., January 2014 <https://www.alienvault.com/blogs/security-essentials/open-source-intrusion-detection-tools-a-quick-overview>
5. Roesch, Martin, “Snort – Lightweight Intrusion Detection for Networks”, Proceedings of LISA ’99: 13th Systems Administration Conference, November 7-12, 1999 <http://static.usenix.org/publications/library/proceedings/lisa99/full_papers/roesch/roesch.pdf>
6. Snort Web Site <https://www.snort.org>
7. Biles, Simon, “Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)”, Computer Security Online, Ltd., 2005 <http://pld.cs.luc.edu/courses/447/sum08/class6/biles.spade.pdf>
8. Nguyen, Thuy T.T.; Armitage, Grenville, “A Survey of Techniques for Internet Traffic Classification Using Machine Learning”, IEEE Communications Surveys & Tutorials, Volume 10, Issue 4, Fourth Quarter 2008
9. Ryan, Jake; Lin, Meng-Jang; Miikkulainen, Risto, “Intrusion Detection with Neural Networks”, AAAI Technical Report WS-97-07, 1997 <https://www.aaai.org/Papers/Workshops/1997/WS-97-07/WS97-07-013.pdf>
10. Beach, Aaron; Modaff, Matthew; Chen, Yan, “Network Traffic Anomaly Detection and Characterization”, Northwestern University, 2010 <http://cs.northwestern.edu/~ajb200/anomaly%20detection%20paper%201.0.pdf>
11. Hong Huang, Hussein Al-Azzawi, and Hajar Brani, “Network Traffic Anomaly Detection”, New Mexico State University, February 2014 <https://arxiv.org/pdf/1402.0856.pdf>
12. Mahoney, Matthew V.; Chan, Philip K., “PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic”, Florida Institute of Technology Technical Report CS-2001-04, 2001 <https://cs.fit.edu/~mmahoney/paper3.pdf>
13. Mahoney, Matthew V., “Network Traffic Anomaly Detection Based on Packet Bytes”, Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 346-350, 2003 <https://dspace-test.lib.fit.edu/bitstream/handle/11141/113/cs-2002-12.pdf?sequence=1>
14. Kim. Seong Soo; Reddy, A.L. Narasimha, “Statistical Techniques for Detecting Traffic Anomalies Through Packet Header Data”, Journal IEEE/ACM Transactions on Networking, Volume 16, Issue 3, June 2008
15. Frenzel, Lou, “What’s the Difference Between the OSI Seven-Layer Network Model and TCP/IP?”, Electronic Design, October 2013 <http://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-osi-seven-layer-network-model-and-tcpip>