www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 4 Issue 5 May 2015, Page No. 11850-11853

Intrusion Detection in Heterogeneous Wireless Sensor Networks using Multi-layer Multi Detector Approach

Ms.Smita H.Karande[1], Mr.S.P.Kosbatwar[2]

[1]Computer Department, SKNCOE,
Savitribai Phule Pune University,
India,
smita.karande86@gmail.com

[2]Computer Department, SKNCOE,
Savitribai Phule Pune University,
India
spkcse@rediffmail.com

Abstract: ***In a WSN, there are two ways to find out inconsistency of object (i.e., an outlier) single-sensing find threat and multiple-sensing find threat. In the single-sensing find threat, the outlier can be successfully find threat by a single user sensor node. On the area, in the multiple-sensing find threat , the outlier can only be find threat by multiple cooperating user sensor nodes .In some applications, the sensed information provided by a single user sensor node might be poor for recognizing the attacker. It is because particular user sensor nodes can only sense a portion of the outlier. For example, the location of an outlier can only be determined from at least three user sensor nodes sensing. The goal of using a layered interface model is to minimize computation and the overall time required to find irregular events. The time required to find anomaly an intrusive event is important and can be minimized by filtering the communication overhead among different layers. This can be achieved by making the layers independent and self-handle to manage to block an attack without the need of a central decision-maker. Every layer in The MLMD framework is learn and adopt separately and then deployed sequentially. We planned 4 protocols that related to the four attacks groups mentioned in the map data.***

Keywords: WSN, Network based, Intrusion find threat, Host based intrusion find threat, Networking.

1. Introduction (Multilayer Multi find Threatner)

We now describe the Multi-Layer multi find threaten (MLMD) in detail. The MLMD draws its motivation from what we call as the Airport Outlier analyzer model, where a number of security checks are performed one after the other in a sequence. Similar to this model, the MLMD represents a sequential Layered interface Protocol and is based on ensuring accessibility, confidentiality, and integrity of data and (or) services over a network.

The aim of using a layered interface model is to minimize calculation and the total time required to find irregular events. The time required to find anomaly an intrusive event is important and can be minimized by filtering the communication overhead among different layers. This can be achieved by making the layers independent and self-handle to manage to block an attack without the need of a central decision-maker. Every layer inThe MLMD framework is learn and adopt separately and then deployed serially. We planned 4 protocols that related to the four attack groups mentioned in the map data.

Each layer is then separately learns and accepts with a small set of related features. Feature selection is important for Layered interface Protocol and discussed in the next section. In order to make the layers independent, some features may be present in more than one layer. The layers basically act as filters that block any abnormal connection, thereby filtering the need of further processing at following layers allowing quick response to intrusion. The effect of such a sequence of layers is that the anomalous events are recognized and blocked as soon as they are find threat.

2. Proposed System Architecture

Proposed System Include Following Stages.

A. Constructing User sensor node Network
B. Data chunk Creation
C. Find authorized and un authorized port
D. Constructing Inter-Domain Data chunk Filters
E. Receiving the valid data chunk

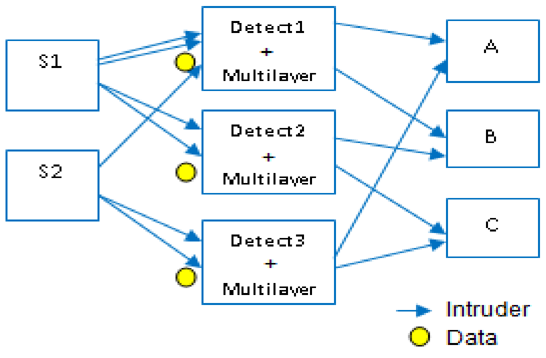


Figure 1: Sending Packets from Source S to D

A. CONSTRUCTING USER SENSOR NODE NETWORK

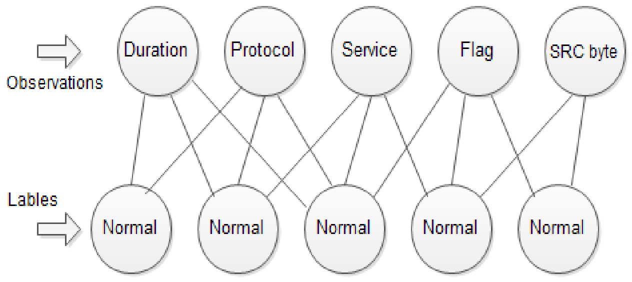
We are going to connect the network .Each node is connected to the adjacent node and it is individually deployed in network area. And also deploy the each port no is authorized in a node.

B. DATA CHUNK CREATION

Browse and select the source file. And approached data is converted into fixed size of data chunks. And the data chunk is send from source to find attacker.

C. CREATE MLMD AND FIND AUTHORIZED AND UN AUTHORIZED PORT

The intrusion find threat is defined as a mechanism for a WSN to find anomaly the existence of wrong, incorrect, or irregular moving enemies. In this module check whether the path is authorized or unauthorized. If path is authorized the data chunk is send to valid destination. Otherwise the data chunk will be deleted. According port no only we are going to find the path is authorized or unauthorized. The MLMD have proven to be very successful in such tasks, as they do not make any unnecessary assumptions about the data. Hence, we discover the suitability of MLMD for intrusion find threat. This System may consider features such as "logged in" and "number of file creations." When these features are learning and summarize individually, they do not provide any information that can aid in find unauthorized user attacks.



However, when these features are learned and summarize together, they can provide meaningful information, which can be helpful for the arrangement task. Taking another example, the connection level feature such as the "service invoked"

Probe layer-Data chunk Feature

The probe attacks are aimed at obtaining information about the target network from a source that is frequently external to the network. Hence, basic connection level features such as the "duration of connection" and "source bytes" are important while features like "number of files creations" and "number of files accessed" are not expected to provide information for find unauthorized user probes.

DoS layer-Traffic Feature

For the DoS layer, traffic features such as the "ratio of connections having same destination host and same service" and data chunk level features such as the "source bytes" and "ratio of data chunks with errors" are significant. To find anomaly DoS attacks, it may not be important to know whether a user is "logged in or not."

R2L layer –Network Feature

The R2L attacks are one of the most difficult to find anomaly as they involve the network level and the host level features. We hence approached both the network level features such as the "duration of connection" and "service requested" and the host level features such as the "number of failed login attempts" among others for find unauthorized user R2L attack.

U2R layer (User to Root attacks)

The U2R attacks involve the semantic details that are very difficult to capture at an early stage. Such attacks are often feature based and target an application. Hence, for U2R attacks, we approached features such as "number of file creations" and "number of shell prompts invoked," while we ignored features such as "protocol" and "source bytes."

D. CONSTRUCTING INTER-DOMAIN DATA CHUNK FILTERS RECEIVING THE VALID DATA CHUNK

If the data chunk is received from other than the port no it will be filtered and discarded. This filter only removes the unauthorized data chunks and authorized data chunks send to destination.

E. RECEIVING THE VALID DATA CHUNK

After filtering the invalid data chunks all the valid Data chunks will reach the destination.

Algorithm & Feature learning:

Step 1: Select the number of layers, n, for the complete system.
Step 2: Separately perform features selection for each layer.
Step 3: Train a separate model with MLMD for each layer using the features approached from Step 2.
Step 4: Plug in the learn and adopt models sequentially such that only the connections labeled as normal are passed to the next layer. Testing
Step 5: For each (next) test instance perform Steps 6 through 9.
Step 6: Test the instance and label it either as attack or normal.
Step 7: If the instance is labeled as attack, block it and identify it as an attack represented by the layer name at which it is find threat and go to Step 5. Else pass the sequence to the next layer.
Step 8: If the current layer is not the last layer in the system, test the instance and go to Step 7. Else go to Step 9.
Step 9: Test the instance and label it either as normal or as an attack. If the instance is labeled as an attack, block it and identify it as an attack corresponding to the layer name.

3. Mathematical VAN Diagram

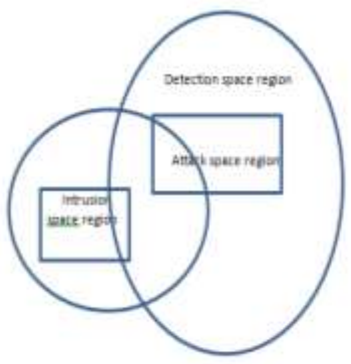


Figure 2: Mathematical VAN

Given mathematical model respect to Venn diagram defines three different and adoptively overlapping space regions: the attack space region (ASR), the intrusion (ISR), and the detection space region (DSR). The attack space region is the space region of all the attacks, i.e., the set of network traces containing an attempt to violate one or more security policies. Whenever the attack is successful, the point belongs also to the intrusion space region (so, ISR ⊆ ASR). Finally, the detection space region contains the network mapped that the outlier i.e. intrusion detection system considers malicious. Due to the inaccuracy of the models, this space region can be considerably apart from the previous ones. Since most of the intrusion detection systems do not take into account the success or failure of an attack (i.e., do not distinguish between attacks and intrusions), usually only four classes of intersection are analyzed:

1. Observed Undetected attacks (p ∈ ASR, p /∈ DSR) are false negatives map
2. Alerts that do not related to an attack (p /∈ ASR, p ∈ DSR) are false positives map
3. Identified true detected attacks (p ∈ ASR, p ∈ DSR) are true positives
4. Usual events with no alerts (p /∈ ASR, p /∈ DSR) are true negative

4. Results

Multilayer-based network intrusion detection systems (NIDS) scan network packets for matches against a known set of intrusion multilayers. Network packets are processed in parallel and go through three phases: capture, reassembly, and detection. The main data structure in the capture phase is a simple FIFO queue, and the reassembly phase uses a dictionary (implemented by a self-balancing tree) that contains lists of packets that belong to the same session, the capture and reassembly phases are each enclosed by protocol stack transaction. Hence, the code for each phase is as simple as that with coarse-grain locks but hopefully achieves good performance through optimistic concurrency. When operating on these data structures, this benchmark has relatively short protocol stack transaction. It also has moderate to high levels of contention depending on how often the reassembly phase rebalances its tree. Overall, since two of the three phases are spent in protocol stack transaction, this benchmark has a moderate amount of total transactional execution time.

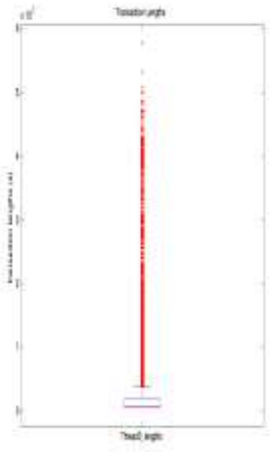
For the test runs, the following parameters were used.

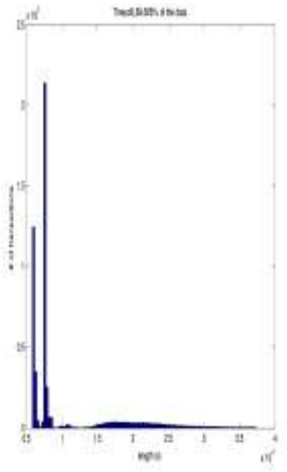
- Percentage of attacks : 10
- Maximum number of packets per stream : 16
- Total number of streams : 20000
- Random seed : 1

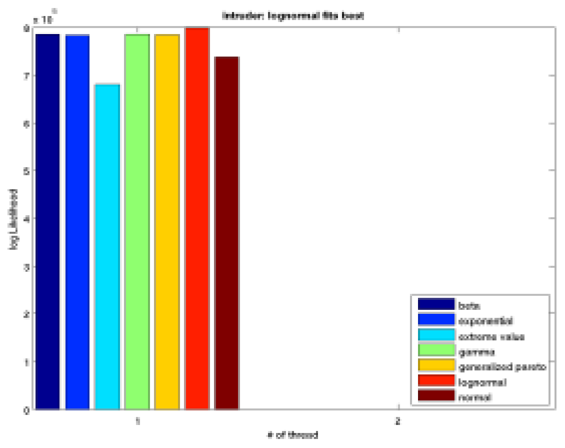
I) Boxplot of entire distribution for 1 threat

II) Histogram showing majority lengths of protocol stack transaction for 1 threat (exact percentage in title of graph)

III) Bar graph of log-likelihood values for 1 threat



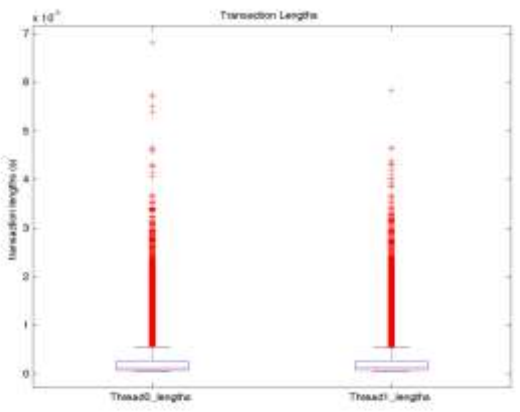


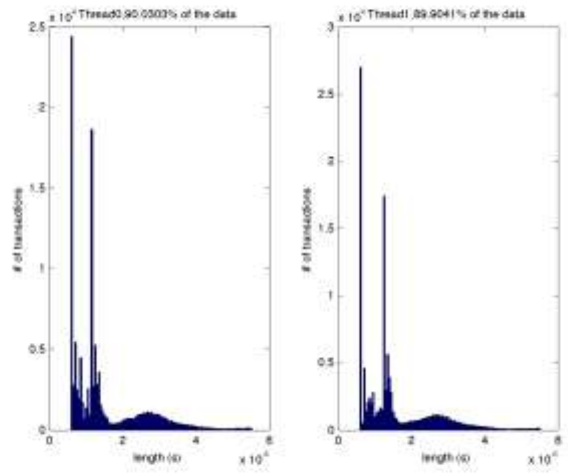


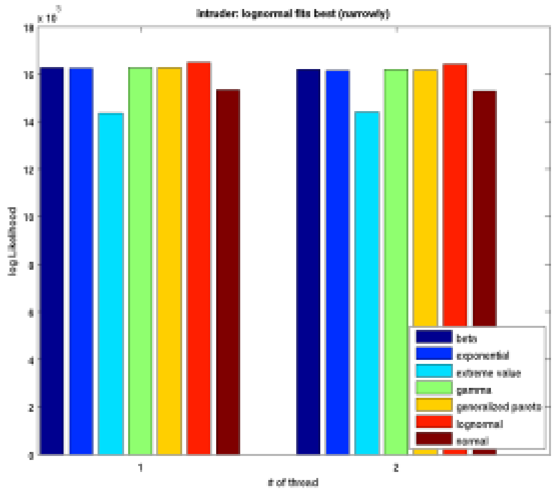
I) Boxplot of entire distribution for 2 threats

II) Histogram showing majority lengths of protocol stack transaction for 2 threats (exact percentage in title of graph)

III) Bar graph of log-likelihood values for 2 threats







Conclusion

This paper analyzes the intrusion find threat problem by describing intrusion find threat probability with respect to the intrusion distance and the network parameters (i.e., node density, sensing range, and transmission range, Percentage of attacks ,Maximum number of packets per stream , Total number of streams).The analytical model for intrusion find threat allows us to logically express intrusion find threat probability within a certain intrusion distance under various application conditions.

References

Write few referred references here (in IEEE format). One such template is given here under.

[1] R. Hemenway, R. Grzybowski, C. Minkenberg, and R. Luijten, “Optical-data chunk-switched interconnect for supercomputer applications,”OSA J. Opt. Netw., vol. 3, no. 12, pp. 900–913, Dec. 2004.

[2] C. Minkenberg, F. Abel, P. Müller, R. Krishnamurthy, M. Gusat, P.Dill, I. Iliadis, R. Luijten, B. R. Hemenway, R. Grzybowski, and E.Schiattarella, “Designing a crossbar scheduler for HPC applications,”IEEE Micro, vol. 26, no. 3, pp. 58–71, May/Jun. 2006.

[3] E. Oki, R. Rojas-Cessa, and H. Chao, “A pipeline-based protocol formaximal-sized matching scheduling in input-buffered switches,” IEEE Commun. Lett., vol. 5, no. 6, pp. 263–265, Jun. 2001.

[4] C. Minkenberg, I. Iliadis, and F. Abel, “Low-latency pipelined crossbar arbitration,” in Proc. IEEE GLOBECOM 2004, Dallas, TX, Dec. 2004, vol. 2, pp. 1174–1179.

[5] C. Minkenberg, R. Luijten, F. Abel, W. Denzel, and M. Gusat, “Current issues in data chunk switch design,” ACM.

[6] C. Minkenberg, F. Abel, P. Müller, R. Krishnamurthy, and M. Gusat,“Control path implementation of a low-latency optical HPC switch,” inProc. Hot Interconnects 13, Stanford, CA, Aug. 2005, pp. 29–35.

[7] C.-S. Chang, D.-S. Lee, and Y.-S. Jou, “Load-balanced Birkhoff-von Neumann switches, part I: One-stage buffering,” Elsevier Comput.Commun., vol. 25, pp. 611–622, 2002.

[8] A. Tanenbaum, Computer Networks, 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1996.

[9] R. Krishnamurthy and P. Müller, “An input queuing implementation for low-latency speculative optical switches,” in Proc. 2007 Int. Conf.Parallel Processing Techniques and Applications (PDPTA’07), Las Vegas, NV, Jun. 2007, vol. 1, pp. 161–167.

[10] H. Takagi, Queueing Analysis, Volume 3: Discrete-Time Systems. Amsterdam: North-Holland, 1993