Towards Adaptive Big Data Cyber-attack Detection via Semantic Link Networks

George Karabatis[1], Jianwu Wang[1], Ahmed AlEroud[2]

[1] *Department of Information Systems, University of Maryland, Baltimore County (UMBC), Baltimore, MD, USA*
[2] *Department of Computer Information Systems, Yarmouk University, Irbid, Jordan*
{georgek, jianwu, ahmed21}@umbc.edu

***Abstract*— As a core mechanism for cybersecurity, the ability to detect cyber-attacks is increasingly critical nowadays. There have been many types of network intrusion detection approaches, such as flow-based and packet-based, targeting single attack and multistage attack detection. Each approach has its own advantages and disadvantages. In this paper, we design an organic combination of these types of efforts into one comprehensive system. Furthermore, to deal with increasing volumes of network traffic and improve full packet analysis efficiency, we employ Spark Streaming platform for parallel detection.**

Keywords*— *Adaptive Cyber-attack Detection; Semantic Link Network; Big Data Platform; Streaming Data Analysis

I. INTRODUCTION AND BACKGROUND

Modern Intrusion Detection Systems (IDSs) analyze the content of network packets to predict attacks. The current approaches for intrusion detection and prediction of network traffic can be categorized differently based on different criteria. We have packet and flow detection based on the data content to be inspected, and single attack and multistage attack detection based on the nature of targeted attacks. We will briefly explain these approaches and try to build a new system combing these efforts.

A. Flow-based and Packet-based Intrusion Detection

The problem of intrusion detection and prediction has exacerbated with increasing volumes of network traffic and IDSs have a hard task analyzing network packets on the fly to provide timely prediction [1]. As a result, there have been attempts to investigate network flows which contain much less information (since it is in an aggregate form) compared to packets in order to predict attacks. The advantage of this approach is that processing smaller amount of information results in higher throughput and faster detection of attacks. On the other hand, its disadvantage is a lower degree of accuracy in detecting attacks due to the lack of payload information.

Flow-based intrusion detection can identify a subset of cyber-attacks, including denial of service [2], scanning attacks [3], worms [4], and botnets [5]. There have been few machine learning techniques utilized in flow-based intrusion detection such as Hidden Markov Models [6] and Support Vector Machines [7] to detect SSH brute force attacks, and entropy [3] to identify anomalies.

Our past work partially alleviates the accuracy problem and increases detection rate by using semantic information that is available in net flows, such as time, location, and other context-related information that are present and links related alerts together. These links are represented in a Semantic Link Network (SLN) which is exploited through an inference process and enhances the detection rate [8].

B. Single and Multistage Attack Detection

While we can detect some types of attacks by only inspecting individual packets/flows, more serious or coordinated attacks might be detected by inspecting all packets/flows with some common features, such as originating from the same IP address, within a longer time period in the past. This type of attack, called *multistage attack*, refers to attacker activities that consist of multiple steps and occur in a certain time [9].

Both flow-based and packet-based approaches have been used for multistage attack detection [10]. Our past work [8] has studied how to detect multistage attacks from sequences of IP flows by efficiently querying the relations produced via reasoning on SLNs. A flow can be predicted as a suspicious activity (that represents a step in a multi-step attack) or a benign activity. During multi-step attacks, several alerts are raised each one representing an indicator of an attack step.

C. Our Approach

In this work, we propose a system design that can support flow-based inspection as initial detection and full packet-based inspection only for quasi-suspicious flows that need further analysis. For full packet analysis, the system will check for both single attacks and multistage attacks. To efficiently process the large volumes of full packets, we employ a Spark platform for data parallelism and streaming data processing. The full packet detection results can be used to update SLNs dynamically. Further, the threshold for SLN-based classification can be automatically adjusted based on factors such as incoming traffic velocity and system workload.

Our contributions include: 1) a hybrid approach that combines a flow-based and a packet-based inspection to detect intrusions; 2) parallel individual and multistage full packet inspection based on Spark Streaming framework; 3) interaction between flow-based and a packet-based inspection for dynamic SLN updates for newly identified attack types and system adaptation based on incoming traffic velocity and system workload.

II. Cyber-attack Detection via Semantic Link Networks

In the following we briefly provide an outline of identifying intrusions based on flows using SLNs. A more detailed explanation of flow-based intrusion detection using context is found in [8].

A *Semantic Link Network (SLN)* is a graph with nodes and edges which are used to infer semantic links [11]. The emphasis in this work is on context, which is used to assist in the process of identifying attacks. Example contextual features of an entity are the time it occurs, its location, the events that target it, and its relationship to other entities [12]. The key idea is that the features which describe a context have a cause/effect relationship to the situation that a specific context may result into, i.e., either an alert or a benign activity.

Net Flows: We use the following definition of a flow structure: (*Isrc*, *Idst*, *Psrc*, *Pdst*, *Prot*, *Pckts*, *Octs*, *Tstart*, *Tend*, *Flags*), where *Isrc* and *Idst* are the features that identify source and destination IP addresses; *Psrc* and *Pdst* are the source and destination ports; *Prot* is the protocol type; *Pckts* and *Octs* give the total number of packets and octets in the data exchange; *Flags* are the TCP header flags; *Tstart* and *Tend* denote the start and end time of the flow respectively.

Constructing SLNs : The SLNs are constructed by generating weighted links among nodes (alert nodes and benign nodes) and then reasoning on such links to augment their semantics. The SLNs are constructed in two major steps: weighted links are created among nodes using similarity; then, reasoning is performed on the links to augment the semantic relationships among nodes. The similarity among nodes is a measure of their co-occurrence. There are three categories of contextual features that have been utilized to calculate similarity. *Time/location*, *numerical*, and *descriptive* features. Time-based features are represented by the timestamp of each alert, the *Tstart*, *Tend* of the flows that contain them and the duration of such flows. Location-based features are represented by the source, destination IPs and port numbers (*Isrc*, *Idst*, *Psrc*, *Pdst*). These features indicate relations among nodes based on source and target of attacks. Numerical features identify traffic statistics such as the number of packets, octets (*Pckts*, *Octs*). Descriptive or nominal features describe other flow characteristics such as the flags and protocol type (*Prot*, *Flags*), in addition to alert description.

For semantic reasoning, initial weights on semantic links among nodes are assigned. The initial weight is the similarity value of time, location, numerical and/or descriptive features. The measures we use to calculate similarity between nodes are Pearson correlation and Anderberg similarity. After the similarity values are calculated, a similarity relationship matrix N is created in modeling the similarity values among nodes.

Flow Prediction Using SLNs: The attack prediction starts by investigating the features of an incoming flow to produce an initial prediction for each flow and pass it to SLNs. This initial prediction is passed to SLNs that expand it to include several other related predictions in the SLNs based on a threshold ∂ which controls the scope of the expansion.

Discarding Possible False Predictions: It is possible to have a scenario where the set of predictions for a specific flow includes both suspicious and benign activities. It is then necessary to discard possible inaccurate predictions (i.e. false positives and false negatives). A second decision tree classification model is created to examine flow features to identify benign activities, resulting in a set of rule-based profile filters that define benign activities. These profile filters are only applied to flows for which the predictions produced include both suspicious and benign nodes, and the result is to identify and remove false positives and false negatives.

III. Hybrid Cyber-attack Detection

This section discuss a novel hybrid approach that combines a flow-based and a packet-based inspection to detect intrusions. Flow-based intrusion detection is fast, yet not so accurate. Packet-based intrusion detection is resource consuming but more accurate. Therefore, a combination of the above techniques is a promising approach that utilizes the advantages of both flow-based and packet-based techniques.

The main and novel idea of this approach consists of two layers: First, a flow-based approach is applied, where the incoming flows are analyzed based on the SLN technique described above and a prediction is made. If the prediction results in a benign flow, then it is allowed to pass into the network environment. Otherwise, the flow that has been characterized as suspicious is further analyzed based on the level of suspiciousness. That is, if the flow was deemed suspicious with a high probability then we follow the policy established for suspicious flows. Usually policies for suspicious flows include denial of entry into the organization's network (reject), diversion into a closely watched subsystem (e.g., diversion to a honeypot), etc.

Apart from these two clear cases, we may encounter flows with a probability of being suspicious in a medium range when we cannot fully identify them as benign or suspicious. For such cases, we delegate the flows into a further more detailed examination with more information from the individual and original packets that the flows were made from. These packets are passed to the second layer of our system, for payload examination. This more thorough investigation of a limited set of packets (just those that comprise the questionable flow) may predict more precisely the outcome of the flow.

Advantages of the hybrid approach are: i) Flows that are predicted as benign or suspicious with high probability do not reach the second layer (packet examination) saving computational resources; ii) Only questionable flows are further examined at the packet level; iii) Accuracy of the prediction is expected to rise, since more information (payload) is available; iv) More attacks may be recognized (since there is access to payload, in addition to flow data); v) Compared to packet based approaches, our approach requires less computational resources.

IV. From Hybrid Detection to Adaptive Big Data Cyber-attack Detection

A main disadvantage of last section's hybrid approach is the payload examination maybe time consuming. To alleviate the problem, we are utilizing Spark Big Data platform for parallel

individual and multistage full-packet analysis. Further, we study how a flow-based component and a packet-based component can interact to achieve better overall performance.

A. Parallel Straming Data Processing

To perform packet-based intrusion detection and prediction for large-scale network traffic streaming data, our new design is on top of the *Spark Streaming* framework[1]. It is an extension of the core Spark API and enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Resilient Distributed Dataset (RDD) is a core abstraction of Spark, which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. A common RDD data type is *key-value* pair where 'key' is used for data partitioning, grouping and aggregation, 'value' contains the partitioned data associated with the key. RDD parallel operations include MapReduce [13], CoGroup, Join and others. Following functional programing principle [14], each RDD operation is a high-level function to run a user-defined function against each element of the dataset. On top of RDD, Spark Streaming provides another abstract called *DStream* which is a sequence of RDDs received within a certain time interval. So the incoming network traffic data can be modeled as continuous DStreams. DStream supports similar RDD operations to continuously process incoming data in parallel in a distributed environment. The main reason we choose Spark is its capability to seamlessly combine different types of data processing tasks, which is needed for our cyber-attack detection application. We can build our application using Spark to support real-time streaming process, database storage and machine learning tasks. Details of Spark and Spark Streaming framework can be found in [15].

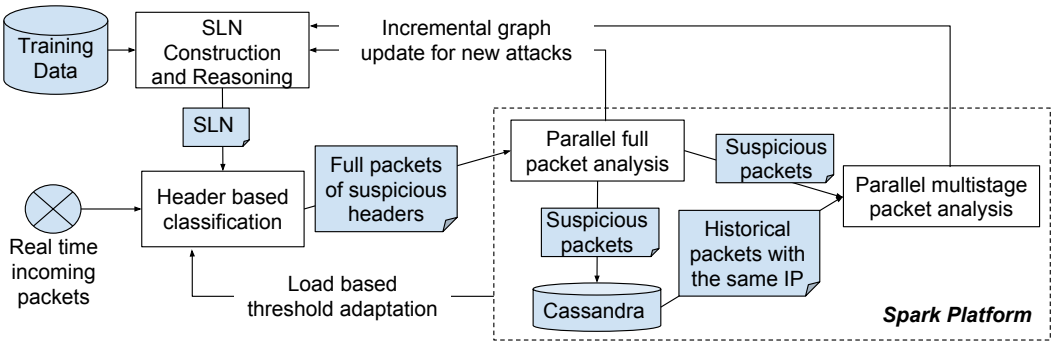

Fig. 1: A Hybrid Big Data Cyber-attack Detection System.

TABLE I. FULL PACKET ANALYSIS ALGORITHM ON SPARK STREAM

1: Create a Spark streaming context with batch interval as *n* second;

2: Create DStream by collecting incoming network socket, a DStream contains all packets within the batch interval time window;

3: Apply full packet analysis function for each packet in parallel through the DStream's map function, output each suspicious packet and its attackType using key-value structure <*packetID*, <*packet*, *attackType*>>;

4: Report all newly identified attacks to update SLN;

5: Set DStream element's key to be IP address in parallel through DStream's map function, output using key-value structure: <ipAddress, <packet, attackType>>;

6: Save suspicious packets into a NoSQL database using key-value structure: <*ipAddress*, <*packet*, *attackType*>>;

7: Load all historical packet contents from the NoSQL database for each IP address in DStream element's key in parallel through DStream's map function, output using key-value structure: <*ipAddress*, {<*packet1*, *attackType1*>, <*packet2*, *attackType2*>, ...}>;

8: Apply multistage packet analysis function for each DStream element in parallel through DStream's map function, output each suspicious multistage packets and its attackType: using key-value structure: <*ipAddress*, <*packet1*, *packet2*, ...>, *multiStageAttackType*}>;

B. Adaptive Big Data Cyber-attack Detection System Design

As shown in Figure 1, our proposed design includes four parts: i) We first use SLN-based detection for all incoming packets and then perform full packet analysis only for the packets whose headers are classified as suspicious by the SLN detection. The hybrid detection can balance the efficiency using the SLN analysis and accuracy using full packet analysis. ii) We utilize Spark platform to achieve parallel full packet analysis since packets can be easily partitioned and analyzed in parallel within a distributed environment. Further, we separate analysis tasks for single attacks and multistage attacks. iii) New attacks identified by the parallel full packet analyses are incrementally added to the SLN graph. The updated new SLN graph can be instantly used for future detections. iv) To adapt the dynamic characteristics of incoming packets in terms of velocity, benign/suspicious packet ratio, we are designing an algorithm to

[1] Spark Streaming framework, http://spark.apache.org/streaming/

set the threshold for SLN-based classification dynamically. The goal is to perform more thorough and accurate detection when incoming packets are in low speed and more timely and swift detection when packets are arriving in high or increasing speed. The algorithm will consider the current load and latency of the full packet analysis, the changes of incoming packet speed and benign/suspicious packet ratio.

Table I shows the main steps for full packet detection process. The basic data type is Spark DStream. We use Map function in Step 3 and 8 for parallel individual and multistage full packet analysis. Multistage full packet analysis needs not only the current packets, but also historical packets with the same IP address. So we use a NoSQL database [16], such as Cassandra to store suspicious packets. We choose NoSQL database because it can store key-value data model directly and many NoSQL databases have good scalability in distributed environments. In Step 6, we use IP address as key to save all historical suspicious packets. Map function is used again in Step 5 to shuffle data using IP addresses as keys, and in Step 7 to read historical packets from the NoSQL database.

V. Conclusion

In this work, we design a comprehensive system to combine different types of approaches for intrusion detection and prediction: including flow-based inspection, full packet-based single attacks and multistage attack detection. Our goal is to study how these approaches can work coherently, especially when they are integrated with Big Data platforms like Spark. We are currently working on the system's implementation and evaluation.

References

[1] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 12, no. 3, pp. 343-356, 2010.

[2] B. Claise. (2008, 24/11/2013). Specification of the Ip Flow Information Export (Ipfix) Protocol for the Exchange of Ip Traffic Flow Information. Available: http://www.ietf.org/rfc/rfc5101.txt

[3] A. Wagner and B. Plattner, "Entropy Based Worm and Anomaly Detection in Fast IP Networks," in 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise., Modena, Italy, 2005, pp. 172-177.

[4] F. Dressler, W. Jaegers, and R. German, "Flow-Based Worm Detection Using Correlated Honeypot Logs," in ITG-GI Conference on Communication in Distributed Systems(KiVS), 2007, pp. 1-6.

[5] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "Botminer: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection," in Proceedings of the 17th conference on Security(USENIX'08), San Jose, CA, 2008, pp. 139-154.

[6] A. Sperotto, R. Sadre, P. Boer, and A. Pras, "Hidden Markov Model Modeling of Ssh Brute-Force Attacks," in Proceedings of the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM '09), Venice, Italy, 2009, pp. 164-176.

[7] P. Winter, E. Hermann, and M. Zeilinger, "Inductive Intrusion Detection in Flow-Based Network Data Using One-Class Support Vector Machines," in 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS'11), Dubai, UAE, 2011, pp. 1-5.

[8] A. AlEroud, G Karabatis. "Context Infusion in Semantic Link Networks to Detect Cyber-attacks: A Flow-Based Detection Approach" IEEE International Conference on Semantic Computing (ICSC) 2014, pp. 175-182.

[9] Alserhani, F., Akhlaq, M., Awan, I.U., Cullen, A.J. and Mirchandani, P., 2010, April. MARS: multi-stage attack recognition system. In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on (pp. 753-759). IEEE.

[10] Zhou, C.V., Leckie, C. and Karunasekera, S., 2010. A survey of coordinated attacks and collaborative intrusion detection. Computers & Security, 29(1), pp.124-140.

[11] Z. Hai, S. Yunchuan, and Z. Junsheng, "Schema Theory for Semantic Link Network," in Fourth International Conference on Semantics, Knowledge and Grid(SKG'08), Beijing, 2008, pp. 189-196.

[12] A. Zimmermann, A. Lorenz, and R. Oppermann, "An Operational Definition of Context," in Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context (Context'07), Roskilde University, Denmark, 2007, pp. 558-571.

[13] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[14] P. Wadler, The essence of functional programming. In Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (pp. 1-14), 1992, ACM.

[15] Karau, H., Konwinski, A., Wendell, P. and Zaharia, M., 2015. Learning Spark: Lightning-Fast Big Data Analysis. O'Reilly Media, Inc.

[16] R. Cattell, Scalable SQL and NoSQL data stores. ACM SIGMOD Record, 39(4), pp.12-27, 2011.