Paper Review

Beehive: Large-Scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks

Chandrika Mukherjee
email: cmukherj@purdue.edu
Date: 17th September 2021

Summary

The paper primarily addresses the issue caused by automatically mining and extracting knowledge from logs, produced by assortment of security products in an enterprise setting. The paper mentions the significance of these logs in detecting potential security threats. To protect their privacy, an organization uses different security products. But due to inconsistency in installation and management, the logs generated by these devices vary largely from each other. The authors have demonstrated a novel idea - "Beehive" to monitor and build useful information out of these "dirty" logs. The paper presents three layers of Beehive system - 1) filter and normalize logs 2) generate features from the normalized log and then 3) applying clustering algorithm to detect suspicious activities by hosts. The authors also provided experimental results on a large scale organization log, and were successful in detecting suspicious activities which could have otherwise gone unnoticed by traditional methods. Beehive reports potential security incidents based on behavioral analysis which improves on traditional signature based methods. However, in my view, the implementation of Beehive is not generalized and is computation heavy. The authors could have utilized semi-supervised learning using the labels provided by security team in the organization. This integrated approach, taken by the authors is however highly relevant in today's world and opens a new research direction in "big-data" security analytics.

Detailed Comments

- In my view, the research work in this paper is highly relevant in today's world. As mentioned on page 1, "administrators' visibility into the posture and behavior of hosts is eroding as employees increasingly use personal devices for business applications." Working from home situation strengthens the purpose of this investigation in the enterprise setting. The primary contribution of this paper is in resolving the real-world challenges of "big data" in security analytics. Unlike previous work, to analyze malicious activities on the network or to detect compromised hosts, the authors didn't use the network data. Instead, Beehive used disparate dirty logs from Security Information and Event Management(SIEM) products in an enterprise setting.

- The paper elaborates on the sanitization and normalization of the logs extensively. The paper explains the necessity to normalize the **timestamps** correctly - in a global network, logs generating devices are kept at different geographical locations. Beehive tries to resolve this problem by utilizing the central SIEM system as a log manager. When the device sends a log to central SIEM, Beehive calculates the time difference of SIEM and the device for a period of time to calculate the time correction value later, which is then used to estimate the normalized timestamps correctly. ($\Delta_{correct} + t_{device_i} = t_{normalized_i}$)

- The authors address the obstacle of dynamic mapping of IP addresses to hosts for a short period of time. Beehive tries to resolve this issue by maintaining hosts to IP mappings like "{IP address, hostname, MAC address, start-time, end-time}" using the DHCP logs collected from SIEM. Beehive utilizes the IP-to-host mappings to generate the static IP address list by getting the difference from total active IPs within the enterprise. Though the bootstrap algorithm described is effective in obtaining the static hosts, the time complexity is high. Some optimization techniques could be used here to make the system behave closer to real-time systems. Beehive utilizes the Windows domain controller log to estimate the dedicated user for a host. The source of gathered data is thus correct and authentic.

- The threat model of Beehive targets two major scenarios - to detect host which shows malicious activity and the hosts who violate the policies set by the organization. The paper has considered 15 features of four categories to accomplish the goal. a) New or unpopular destinations, b) Software configuration of hosts, c) Company policies, and d) Traffic volume and time.

  - The paper demonstrated building a history of external destinations reached by the hosts over time, later used the history data to identify if a host is connecting to an unpopular site. The authors also improve the time and space complexity of the algorithm by introducing custom whitelisting, filtering, and domain folding. An external destination is included in the whitelist if interaction with internal hosts exceeds a threshold. The paper shows a 74% reduction in log processing by applying this improvement. Although the numeric figure is promising, as the number of new destinations does not decrease over time, the history storage will continue to increase indefinitely. Therefore, in my view, the implementation is still not well suited for real-world application.
  - The paper correctly considers the fact that "Users are most commonly directed to new sites by search engine, news sites, or advertisements." Destinations that are not referred to by whitelisted sources are therefore rightly considered suspicious.

  - The paper utilizes the fact that the hosts within an organization have similar software configurations. Beehive uses the user agent (UA) string in the HTTP request header to determine if the application that the user has requested is safe or not. UA also informs about the operating environment.
  - The authors considered the policy violation of the enterprise as an indication of suspicious activity that addresses the second part of the threat model. Additionally, Beehive monitors the traffic volume, if within a certain time window, a host generates more connections than a threshold, the activity is considered suspicious.
  - In case of feature generation, the paper has considered different static thresholds to decide popularity of destinations, monitoring traffic volume for a period of time. These thresholds are based on experimental data which doesn't indicate their efficacy for logs of different volume in real-time. The authors could have used learning methods on threshold to decide the value of threshold dynamically.

- The authors mentioned on page 4 that feature selection was based on known malware practices and policy violations within EMC. Although these scenarios are valid indicators of security threats, features will vary for a separate organization. Therefore, in my view, Beehive doesn't achieve generalization.

- The authors have applied unsupervised learning on the features due to the absence of ground truth or labeled data. First, Principal Component Analysis(PCA) is performed on the features to determine major contributing features. Then the K-means algorithm is applied to the data. After one iteration of K-means, most of the hosts fall into a cluster, whereas the behaviors of a few deviate significantly. In my view, the K-means algorithm approach with the limited features was successful for EMC-based experiments because of several constraints in an enterprise environment, and the similarity in the work setting of different hosts. Therefore, detecting outliers was less difficult than of an open internet.

- Although the ground truth was initially unavailable, after the Security Operation Center(SOC) acknowledges or rejects the incidents reported by Beehive, those labeled data could be then used to perform a semi supervised learning. In a semi supervised learning, very small amount of data is labeled and large amount of data is unlabeled. Using unsupervised learning, the structure of input data could be learnt and the supervised learning could be used to guess the best label for unlabeled data. These predictions could be fed again to the training model to predict on new unseen data.

- The experiment at EMC over the period of two weeks reported 748 Beehive incidents and out of which only 8 (1.02%) were detected by state-of-the-art security tools. The paper also mentions that destination-based features were most useful in detecting dynamically generated domains(DGA) malware which could not be detected by antivirus and enterprise's Security-Operation Center(SOC). Beehive also detected unusual activities of automated applications. These shreds of evidence insinuate the ability of Beehive to detect previously unknown suspicious behaviors. Thus Beehive's behavioral analysis improved on previous signature-based detection which satisfies the aim of this paper.

Recommendations

The paper makes a great effort in resolving real-world challenges in "big-data" security analytics. Specifically, normalization of timestamp, IP-to-host mapping, custom whitelist creation which reduces the naive log processing by 74%. Not only filtering of unrelated logs from SIEM, but the generation of features from the standardized data also had its challenges. Beehive could detect malware infections, policy violations during the experiment on the large-scale logs of EMC that could not be detected by traditional systems. Organizations use state-of-art security technologies deployed both on network perimeter and hosts, so, the network data, obtained from them lack security threats. Malware is cleaned or quarantined, therefore, using the clean data to detect security events was a challenge to the Beehive system. Thus this research paper plays a significant role in opening a new research direction towards finding security threats using a large volume of disparate log data from enterprise settings to the open internet.

Although it was a novel approach by the authors, the time and space complexity of computation for even filtering and generating features was huge. The experiment was done over a period of two weeks. In real-time the log size will increase indefinitely, therefore, the cost of computation will be even more. The research work is performed in a constrained environment and feature selection was done based on uniformity in an enterprise. So, to extend the scope of Beehive in an open internet, new approaches need to be taken, however, Beehive can work as a building block for the open internet solution. Additionally, a semi-supervised approach could have provided more reliability on the results obtained by Beehive.