December 2022

SELF-CORRECTING POLICIES USING BEHAVIORAL ANALYSIS OF NETWORK SECURITY NODES

Doron Levari

Vincent E. Parla

Tariq Ahmed Farhan

Siddhu Warrier

Jay Perry

SELF-CORRECTING POLICIES USING BEHAVIORAL ANALYSIS OF NETWORK SECURITY NODES

AUTHORS:
Doron Levari
Vincent E. Parla
Tariq Ahmed Farhan
Siddhu Warrier
Jay Perry

ABSTRACT

For every organization, the requirements for network security and access are constantly evolving. The recent pandemic served to accelerate many of those requirements. Additionally, external threats evolve and multiply as well. An automatic system that offers policy-centric insights, anomaly identification, potential courses of action, and remediation recommendations is a key to enable the fast, agile, and accurate policy adjustments that are required to address the above-described security requirements at an increasing pace. Techniques are presented herein that solve the aforementioned problem by applying distributed behavioral anomaly detection that feeds into a centralized policy distribution system to provide a policy self-correction mechanism. Aspects of the presented techniques look into all of the policy and configuration components, such as objects, rules, routing, and more. Further aspects of the presented techniques leverage machine learning (ML) capabilities.

DETAILED DESCRIPTION

Modern security policies are complex and distributed. As a result, it has become extremely difficult to track usage, identify behaviors, remediate incorrect policies, and obtain feedback regarding behavioral changes that may be introduced into a network due to operational (e.g., policy) changes.

Operational changes to policies and configurations in a network are commonplace. However, there is no effective mechanism to provide insights and feedback regarding the operational changes that are made by administrators. In this context, feedback refers to behavioral feedback from the network itself. This missing element is necessary in that the

feedback adds a layer of confidence to the operational changes that are being made (preventing misconfigurations and allowing for self-correction) and supports the maintenance of an audit log of all of the behavioral changes (with an opportunity to correlate the same to causative changes).

Techniques are presented herein that solve the aforementioned problem by applying distributed behavioral anomaly detection that feeds into a centralized policy distribution system to provide a policy self-correction mechanism.

Existing solutions detect anomalies in behaviors, but they do not offer further policy-centric insights into what caused the anomalies, nor do they offer a potential remedial course of action if an anomaly was not expected or is not legitimate.

Aspects of the techniques presented herein look into all of the policy and configuration components, such as objects, rules, routing, and more. Further aspects of the presented techniques maintain an audit log of behavioral changes in a system, as opposed to typical audit logs which record just change events, thus constructing a history of the network which becomes invaluable during incident response. In brief, the presented techniques yield self-correcting policies that are based on the behavioral analysis of network security nodes.

The techniques presented herein encompass a series of steps, which will be described during the next portion of the instant narrative.

During a first step, the behaviors of network nodes may be learned by a centralized policy manager over time using machine learning (ML) techniques. Such an approach may consider the traffic profile with full-stack visibility, from network to applications, to identity a number of elements (including, but not limited to, traffic that is spread over various networks, transports, and applications) both quantitatively and qualitatively. Exemplary elements may include particulars regarding general node health such as central processing unit (CPU) usage, memory consumption, and events data. The above-described information establishes usage patterns and a baseline and learns the behavior that is expected by, for example, an identity, a person, or a department. For example, "access to Facebook by the marketing department during mornings consumes on average 1 gigabyte (GB) of data per hour" is one example of an insight that may be delineated through such learning.

Then, the techniques presented herein may continually keep refining and aligning to the established baseline (as described above), comparing traffic patterns to it, detecting anomalies, and recommending policy changes.

Finally, if any intentional policy changes are made as part of regular operations on a network, or even through malicious rogue agents, and those changes cause the network's nodes to deviate from the desired baseline, the techniques presented herein not only identify the undesired behavioral change in the system but add a feedback loop that offers insights into what policy changes caused the behavior as well as, optionally, automated remediation.

It is important to note that within the techniques presented herein, as described above, the term "policy" encompasses all of the manageable components of a network node, including, for example, all of the configuration information.

Administrators who author change events are typically aware of the expectations from an operational change (i.e., if a behavioral change is to be expected). This is the feedback layer that is lacking in current solutions and which the techniques presented herein automatically provide, corelating behavioral changes to change events (without causation) and offering a probability percentage which may assist in establishing causality. The deep policy and configuration insights that are already available within a system allow for discerning which change event could have potentially caused what behavior change. If a behavioral change is classified as positive by an administrator, then it may be accepted. However, if a behavioral change is classified as negative by an administrator, then the policy may self-correct based on the corelated change events that the administrator confirms.

As described above, the techniques presented herein support an administrator-driven decision-making process. A specific policy change recommendation may be derived from change events in the corresponding timeframe that are related to a specific component within the policy that could have potentially triggered the behavior change along with the identification of new traffic behaviors and applying zero-trust policies to the same. It is important to note how such an approach differs from simple automation. In contrast to the typical elements of automation, aspects of the techniques presented herein learn what change events cause what behaviors, identify subcomponents of a policy that could have

triggered the behavioral change, learn network node behaviors, and provide network behavioral feedback on change events – all of which may be modeled through ML techniques. The audit log of behavioral changes to a network (and its subsequent correlation and potential causation to change events) are the outcomes of an entire system according to the techniques presented herein and allow administrators to make informed decisions in incident response situations. Such an approach results in a better informed "decision support system" for an administrator. Additionally, the Information Technology Infrastructure Library (ITIL) processes of change management and incident management remain similar and are now improved, now better described and documented, and have better empirical justification.

Additionally, the techniques presented herein provide correlation, and offer a probability of causation, but not absolute causation. Causation potentiality may be established through the centralized policy intelligence that a policy manager component within the system possesses where the policy manager has learned the impacts of change events in the past by leveraging, for example, time-based analysis.

Among other thigs, the techniques presented herein encompass a learning phase. Figure 1, below, depicts elements of one such learning phase that is possible according to aspects of the presented techniques and which is reflective of the above discussion.

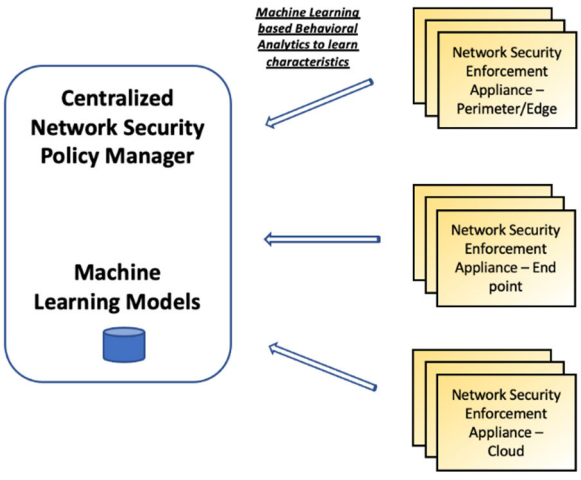

Figure 1: Exemplary Learning Phase

The techniques presented herein may be further explicated with reference to two illustrative use cases. Figure 2, below, illustrates elements of a first use case.

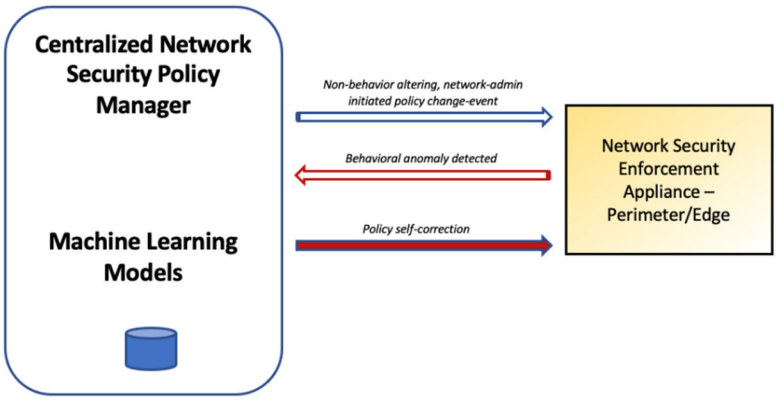

Figure 2: First Illustrative Use Case

As depicted in Figure 2, above, under the first use case an active change event triggered some undesired behavioral change, leading to a subsequent policy correction. Figure 3, below, illustrates elements of a second use case.

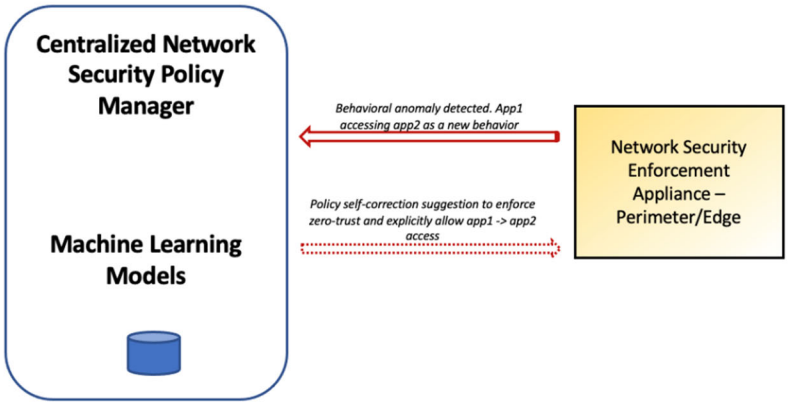

Figure 3: Second Illustrative Use Case

As depicted in Figure 3, above, under the second use case a behavioral anomaly was detected without any change events, leading to the suggestion of a policy correction and thus ensuring zero trust.

Figure 4, below, depicts elements of one possible illustrative system (i.e., a Defense Orchestrator) that may be realized through the use of aspects of the techniques presented herein.

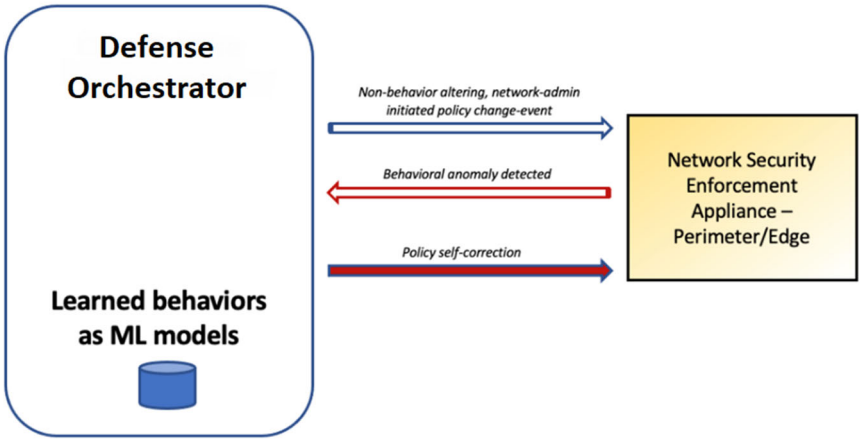

Figure 4: Illustrative System

It is important to note that the techniques presented herein, as described and illustrated in the above narrative, enjoy full-stack visibility and go far beyond the variables that are considered by other solutions. This is allowed for by introducing a system that has complete visibility into the policies and configurations regarding a network (i.e., a full-stack traffic profile from a network layer to a transport layer to an application layer) to identity both qualitative insights (such as, for example, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Stream Control Transmission Protocol (SCTP), Internet Protocol (IP), Transport Layer Security (TLS), etc.) and quantitative insights (such as, for example, connection number, spread, etc.) from different device metrics, including CPU usage and memory consumption, and develop insights into event data.

It is also important to note that the centralized policy manager that was described in the above narrative is a policy-first solution that has enforcement rights. While other solutions employ a segmentation approach, the techniques presented herein principally employ a centralized policy manager that realizes elements of segmentation as a side-effect of solving the problem of a lack of behavioral feedback regarding change events.

In summary, techniques have been presented herein that support the application of distributed behavioral anomaly detection that feeds into a centralized policy distribution

system to provide a policy self-correction mechanism. Aspects of the presented techniques look into all of the policy and configuration components, such as objects, rules, routing, and more. Further aspects of the presented techniques leverage ML capabilities.