J Vis (2019) 22:955–971
https://doi.org/10.1007/s12650-019-00580-7

REGULAR PAPER

Xin Fan · Chenlu Li · Xiaoru Yuan · Xiaoju Dong · Jie Liang

An interactive visual analytics approach for network anomaly detection through smart labeling

Received: 7 July 2019 / Accepted: 18 July 2019 / Published online: 3 September 2019

Abstract Network anomaly detection is an important means for safeguarding network security. On account of the difficulties encountered in traditional automatic detection methods such as lack of labeled data, expensive retraining costs for new data and non-explanation, we propose a novel smart labeling method, which combines active learning and visual interaction, to detect network anomalies through the iterative labeling process of the users. The algorithms and the visual interfaces are tightly integrated. The network behavior patterns are first learned by using the self-organizing incremental neural network. Then, the model uses a Fuzzy c-means-based algorithm to do classification on the basis of user feedback. After that, the visual interfaces are updated to present the improved results of the model, which can help users to choose meaningful candidates, judge anomalies and understand the model results. The experiments show that compared to labeling without our visualizations, our method can achieve a high accuracy rate of anomaly detection with fewer labeled samples.

Keywords Anomaly detection · Interactive labeling · Visual analysis

1 Introduction

With the development of information technology, network has become a necessity of most people's life. However, every coin has two sides, network provides convenience and benefits for people, but it also gives the initiative to malicious attackers who steal private information or make public network facilities paralyzed in order to obtain unlawful economic profits. In recent years, the number and the strength of network

X. Fan · C. Li · X. Dong (✉)
BASICS, Department of Computer Science and Engineering, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
E-mail: xjdong@sjtu.edu.cn

X. Fan
E-mail: fanxin0601@sjtu.edu.cn

C. Li
E-mail: chenluli@sjtu.edu.cn

X. Yuan
Key Laboratory of Machine Perception (Ministry of Education), and National Engineering Laboratory for Big Data Analysis and Application, Peking University, Beijing, China
E-mail: xiaoru.yuan@gmail.com

J. Liang
University of Technology Sydney, Ultimo, Australia
E-mail: jie.liang@uts.edu.au

attacks are gradually increased, and the intrusion methods are also updated, making the network information security problem more and more serious and crucial.

To be able to identify and prevent network attacks and ensure network security, intrusion detection technology has attracted considerable attention over the years. Intrusion detection can be divided into two types: misuse detection and anomaly detection (Mukherjee et al. 1994). Misuse detection is based on the rules or signatures predetermined by the domain experts and cannot detect unknown attacks. Anomaly detection attempts to find patterns in data which do not conform to an expected normal behavior. It is capable of identifying new attacks, but may has a relatively high false detection rate. This paper focuses on anomaly detection, in order to adapt to complex and changeable network attacks.

Despite current research achievements on anomaly detection (Bhuyan et al. 2014; Chandola et al. 2009; Buczak and Guven 2016), there are still some challenges and study directions worth exploring. One of the problems is that the model lack interpretability, which means that the model is a black box for the users, and the results produced are difficult to understand. Some works try to combine visualization technology to alleviate this problem. What's more, most of these works are supervised, and one may not get reliable results without a large amount of labeled data to train models. Nevertheless, applicable public labeled datasets are hard to obtain in many cases. To address this problem, active learning (AL) approaches are often adopted (Görnitz et al. 2009; Almgren and Jonsson 2004). AL queries the most useful unlabeled samples for the current classifier through certain strategies (Settles 2009), ask the experts to making labeling and then use these samples to improve the model. The drawback is that AL do not consider the user's ability of choosing meaningful instances.

This paper proposes a novel smart labeling method. Combining active learning and visual interaction, it helps the analysts in the field of cyber security to label network behavior patterns iteratively, so that the anomalies can be detected. We call it smart labeling because it can automatically expand human labels and give suggestions about the next candidate. Users can choose the candidate instances independently based on the distribution of network behavior patterns as well as the influence degree of instances suggested by the model (Fig. 1). The model will then produce improved labels according to the user's feedback. The experiment shows that our method can improve the efficiency of labeling and increase the detection accuracy of the learning methods by a small number of manual labels compared with AL methods and other automatic anomaly detection methods. Besides, to satisfy the practice requirement, our algorithms are designed to be incremental.

Contributions of our work could be summarized as follows:

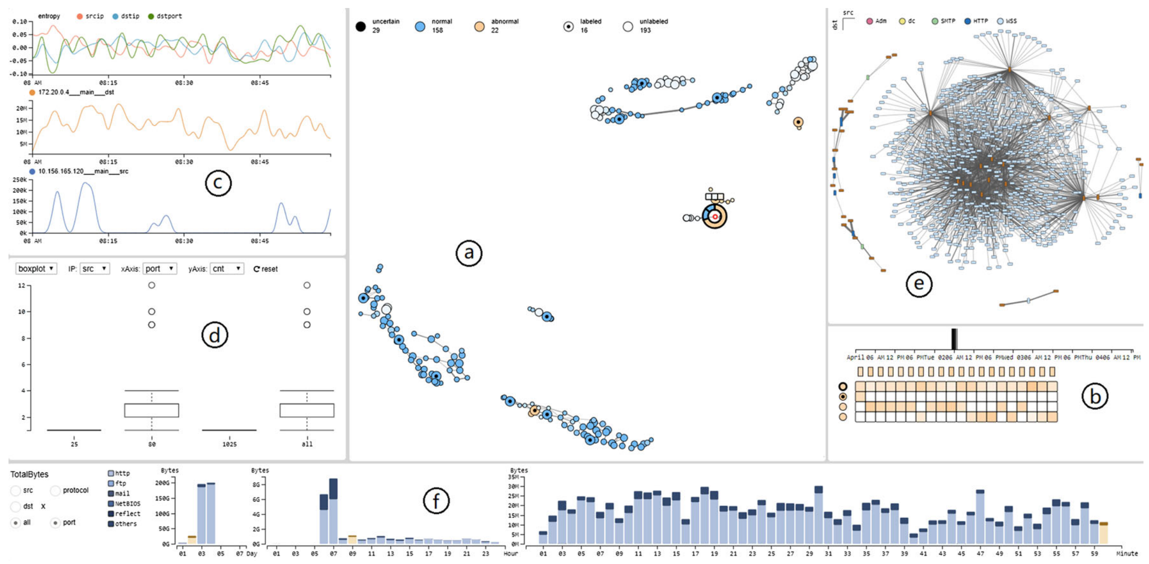


Fig. 1 System overview. **a** The main view shows the network behavior patterns to be labeled. **b** The matrix view helps users to select the time slice and check similar nodes. **c** The line charts show the curves of entropy and traffic flow. **d** The boxplot/scatter view shows some statistics information. **e** The network's topological structure. **f** The timelines for days, hours and minutes

- A new network anomaly detection algorithm based on active learning as well as visual interaction. In comparison with other works, our approach involves user's interactive labeling into the iterative learning process, and regards the anomaly detection problem from the angle of data labeling so that a large number of unlabeled data in the field of network security can be taken into the full advantage. Besides, the model can be adjusted incrementally.
- Novel visual design to guide users to select the instances for labeling. The network behavior patterns, the confidence of the models predictions and the influence degree of instance are visualized in our system. The main view will be updated according to users' feedback and the adjustment of the model. We also provide multiple auxiliary views to help users to judge anomalies.

The rest of this paper is organized as follows: Sect. 2 introduces the most related works to this paper. Section 3 gives an overview of our method. Section 4 describes the algorithms that we used for mining the network data. Section 5 presents the visualization design of this work. In Sect. 6, we analyze the data from the challenge 3 of VAST Challenge 2013 to validate the system. In Sect. 7, we make four experiments. Finally, in Sect. 8, we summarize our work and point out some research directions in the future.

2 Related work

In this section, we review researches that are most relevant to our work in aspects of labeling, network anomaly detection and visualization.

2.1 Network anomaly detection

Ahmed et al. analyzed four major categories of anomaly detection techniques which are classification, statistical, information theory and clustering (Ahmed et al. 2016). Bhuyan et al. categorized existing network anomaly detection methods and systems (Bhuyan et al. 2014).

Statistics and machine learning have been widely used in anomaly detection in recent decades. Buczak and Guven (2016) provided a brief introduction of various models. Sommer and Paxson (2010) provided a set of guidelines for applying machine learning to detect network intrusion. The problem of supervised learning is that there are no labeled datasets for training in the field of network security.

Some unsupervised learning models can also be applied to detect anomalies. Bruns-Smith et al. (2016) used ENSIGN, a tensor decomposition toolbox to discover external attackers, trace the evolution of the attack over time and separate disguised abnormal DNS activity. Chen et al. (2017) provided a new ensemble clustering (NEC) method to detect anomalies. Yousefi-Azar et al. (2017) use deep Auto-encoders (AEs) to learn latent representation of different feature sets. Considering the periodic feature of data, Zhang et al. (2017) present an algorithm called Periodic Self-Organizing Maps to detect anomalies in periodic time series.

Nevertheless, both supervised and unsupervised methods exist the problem of insufficient user engagement. On the one hand, as users are not involved in the learning process, the model cannot be improved or adjusted according to user experience. On the other hand, the interpretability is also insufficient for that users always feel difficult to understand how the models produce the final results.

2.2 Labeling

For the labeling task, two commonly used approaches are AL and visual interactive labeling. The difference between them is that AL is model-centered, while visual interactive method is user-centered. Bernard et al. (2018a) systematically compared the performance of AL and visual interactive labeling.

AL asks human experts for information on selected instances. It can train a relatively good learning model with a small number of labeled instances. The problem is that AL do not consider the user's ability of choosing meaningful instances. Although the users can identify patterns fast, they cannot decide the candidate instances

On the other side, visual interactive method can fully utilize the experience and ability of the experts and can also greatly improve the interpretation of the models (Bernard et al. 2018b). The major drawback is that the biased instance selection of users can result in sub-optimally trained models.

Recent researches tend to combine visualization and AL. Heimerl et al. (2012) presented an approach for building classifiers interactively and visually to label textual documents. Höferlin et al. (2012) used interactive learning in the domain of video visual analytics. RCLens (Lin et al. 2018) allows the user to select features, adjust data labels and the ML model to combine the user's feedback to model. Bernard et al. (2015) present a visual active learning system that enables physicians to label the well-being state of patient histories suffering from prostate cancer.

Although visual active learning has been applied to many areas, it has been little studied in the field of network anomaly detection.

2.3 Network visualization

As a bridge between the users and the model, visualization is an important part of our approach. There has been considerable research in the field of network security visualization. Shiravi et al. (2012) have introduced the categories of network security data as well as network security visualization. IP addresses and ports are indispensable parts of network security analysis. HNMap (Mansmann et al. 2007) uses Treemap to express IP addresses. Force-directed layout algorithm with bundling techniques (Fischer et al. 2008) can also be used to show network topological structures. PortVis (McPherson et al. 2004) uses a 256×256 matrix to express the traffic situation of 65,536 ports and a movable small window to observe detail information. For complex network attacks, radar charts can be used to better identify the association between events. VisAlert (Livnat et al. 2005) uses the 3W (what when where) model and radar chart to analyze network data.

3 Design and overview

The system uses NetFlow data, which combines a series of messages between two computers into a single flow record by using the standard data exchange mode to process the first package.

According to the discussions in the introduction and a review of related works, we describe the most critical requirements (R1–R4) that guide the design of our approach as follows.

- *R1 Human in the loop* As we lack enough labeled data in actual networks, many supervised models no longer apply, and therefore, user experience should be fully utilized, especially in the field of network security. For this purpose, the algorithm and the visualization modules should be tightly integrated.
- *R2 Suggestion and result feedback* The algorithm should provide suggestions based on some strategies for users to select candidate instances. For example, the algorithm can use error reduction schemes and prompt users for instances that will change the model most. Besides, after user labeling, the algorithm should make in-time response to help users to learn how their labels change the classification results.
- *R3 Incremental learning ability* For actual demand, the models should be able to be trained incrementally to learn the new samples. By this means, the costs of retraining model and user labeling can be avoided, and the history data can also be utilized.
- *R4 Situation awareness and anomaly detection* According to the investigation of network security visualization researches, both the overview views for network situation and the detail views for specific information are needed. Users often switch between them to judge abnormal conditions. The suggestions and classification results of the algorithm and the distribution of the network behavior patterns are also needed to be visualized in order to help users understand model and select candidate instances.

Based on the above requirements, the system is mainly composed of feature selection module, network behavior pattern recognition module, pattern classification module and interactive visual interfaces (Fig. 2). First, features are extracted from NetFlow data, and then, an auto-encoder is used to compress features. The compressed features are then fed into SOINN to learn network behavior patterns. These two models can be trained incrementally, which supports R3. Next, an algorithm based on FCM is applied to do classification according to existing labeled instances and then recalculate the influence degree of instances. These results and the distribution of behavior patterns are visualized to assist the users to make decisions (R2). Users can judge whether a pattern is abnormal or not on the basis of the network situation and detail information provided by the visual interfaces (R4). Through this iterative labeling process, the detection rate of anomaly can be increased gradually. The proposed approach keeps users in the analysis loop and allow users to detect and label the anomalies interactively, supporting R1.

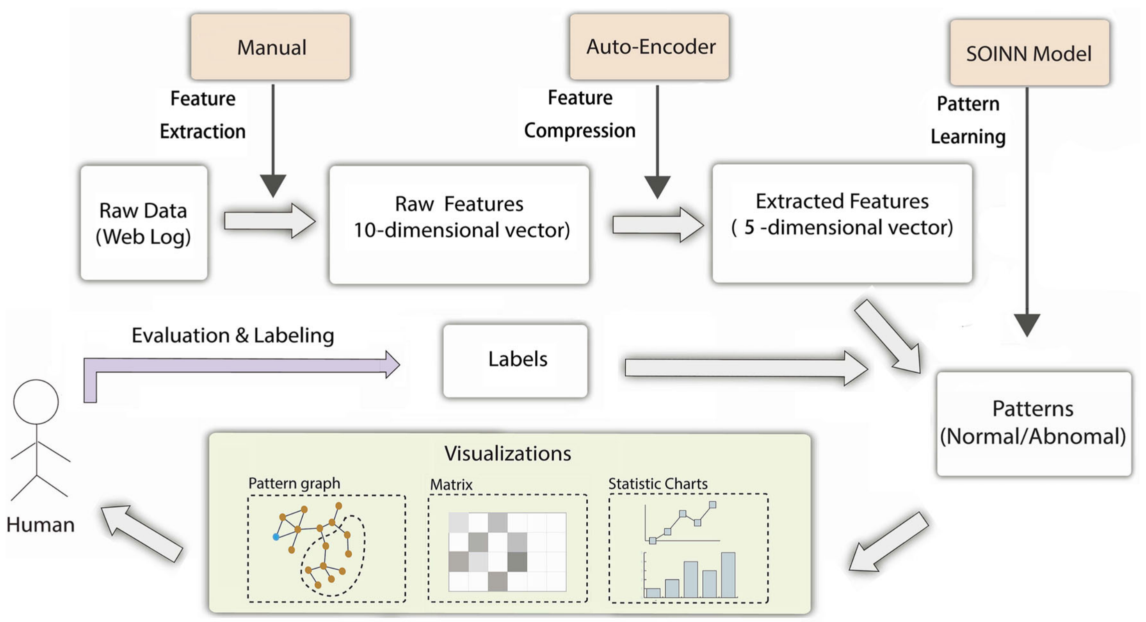


Fig. 2 System flowchart. The network behavior patterns are obtained by using AE and SOINN and are labeled by users through an interactive visual interface and a clustering algorithm. The labeling process is to iteratively check the SOINN topology view, matrix view and detail views

4 Model design

In this section, we describe the model of our system in detail.

4.1 Feature extraction

The system first splits the total NetFlow data into 1-min time slices according to the time stamp and then extracts the following 10 features from each time slice: number of records, number of protocols, entropy of protocols, number of distinct destination IPs, entropy of destination IPs, number of distinct destination ports, entropy of destination ports, average of duration time, average total bytes, average number of packets. These features have maximum information gain in NSL-KDD dataset (Aljawarneh et al. 2018). When network anomaly happens, some of these features may show unusual changes. These features are often used by the researches in the network security area, such as Velea et al. (2017) and Chen et al. (2017).

Specifically, we aggregate the NetFlow data within 1 min into a single vector, and all the feature vectors are ordered in time. Therefore, the sequence of the feature vectors itself indicates the time attributes.

The next step is feature compression. This is necessary to avoid the improper distance measurement of SOINN in the next step, which may be caused by some correlative features. As in Yousefi-Azar et al. (2017), we use AE to learn a type of suitable representation of network data because in contrast to other feature engineering approaches, AE can provide more discriminative features. It uses a nonlinear activation function and can capture the semantic similarity between feature vectors.

AE is a kind of neural network that uses a back-propagation algorithm to make the input data as close to the output data as possible. It compresses the original data into latent representations first and then uses them to reconstruct data.

We feed the extracted features into AE to get the compressed five-dimensional feature vector. We choose five dimensions because the reconstruction error will increase with lower dimension according to test.

4.2 Behavior pattern learning

Although we can map the obtained feature vectors to a two-dimensional plane directly in a similar way as scatter diagram to ask users for labeling, such a method has some problems. On the one hand, the visualization can be cluttered when the raw data covers a long time range. On the other hand, the entity we hope to label is an entire network behavior rather than every 1-min time slice. Usually, a network behavior is composed of many continuous time slices. Therefore, we then aggregate the feature vectors to network behavior patterns.

Topology-based methods are beneficial to explore the relationships between network behavior patterns. They also alleviate the problem of the imbalanced network data. Compared with growing neural gas (GNG) (Fritzke 1995) and self-organizing map (SOM) (Kohonen 1990), SOINN has no fixed network size or learning rate, which has more stable performance and is appropriate for discovering patterns from large-scale data (Sudo et al. 2009).

SOINN is a two-layer competitive neural network. It obtains the online clustering and topological representation of input data in a self-organizing manner (Furao and Hasegawa 2006). The first layer adaptively generate neurons to represent the input data online. The two neurons which are closest to the input vector are declared as winners, and their weights are updated to close to the input vector. The second layer uses the neurons generated in the first layer as input to run the algorithm again to stabilize the learning result. Neurons here can also be called nodes. The nodes generated by SOINN and the connections between them can reflect the distribution and topological structure of the original data.

After the training finishes, each node has attributes such as weight and threshold and can represent a kind of network behavior pattern.

4.3 Pattern classification

Based on the obtained behavior patterns, we can calculate each pattern node's certainty of belonging to various types. Each node has certainty for three types: normal, abnormal and uncertain. The sum of them equals one. Initially, the type of each node is uncertain.

Then, considering FCM (Fuzzy c-means) algorithm (Dunn 1973; Zhao et al. 2018), an unsupervised fuzzy clustering method. The clustering result of the algorithm is the membership grades which indicate the degree to which data points belong to each cluster. This is similar to our demand, so we can use the membership to represent certainty. The FCM aims to minimize an objective function:

$$\sum_{k=1}^{c}\sum_{x_i \in X} u_{ik}(d_{ik})^2 \tag{1}$$

where X is sample set and c is the number of cluster centers.

Use U to represent membership matrix. The ith sample's membership value of belonging to the kth cluster center is:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left[\frac{d_{ik}}{d_{ij}}\right]^{\frac{2}{m-1}}} \tag{2}$$

where m is the hyper-parameter that controls how fuzzy the cluster will be and d_{ik} is the distance between the kth sample and the ith cluster center. Repeat the calculation of cluster centers and membership matrix until the algorithm has converged.

We apply FCM to our system, as described in Algorithm 1.

Algorithm 1 Certainty Calculation

```
1: Initialize global cluster centers A={$s_i$} by users labeled patterns, i=1,2,...,C. C is the number of cluster centers, N is the number of patterns. Each pattern has certainties for three types: Type={normal, abnormal, uncertain}.
2: Labels propagation. For each patterns $s_i$, if it is within the threshold of some cluster centers, suppose the set of these cluster centers is $A_{s_i}$, update the certainty of $s_i$.
   $certainty_{s_i,k} = \frac{\sum_{j \in A_{s_i}} d_{ij} - \sum_{j \in A_{s_i}, type_j = k} d_{ij}}{\sum_{j \in A_{s_i}} d_{ij}}, \forall k \in type$
3: Add local cluster centers and calculate certainty. Suppose $N_{s_i}$ is the set of the neighbors of $s_i$. The certainty of $s_i$'s belong to type k $certainty_{s_i,k}$ is:
   $\sum_{\substack{s_m \in A \\ type_m = k}} \frac{1}{\sum_{j=1}^{C} [\frac{d_{im}}{d_{ij}}]^2} + \sum_{s_m \in N_{s_i}} \frac{1}{\sum_{j=1}^{C} [\frac{d_{im}/certainty_{s_m,k}}{d_{ij}}]^2}$
4: Normalization to calculate the type certainty.
   $certainty_{s_i,k} = \frac{certainty_{s_i,k}}{\sum_{k \in Type} certainty_{s_i,k}}$
   $i = 1, 2, \ldots, N, \forall k \in Type$
5: Calculate the change of certainty. If the change is less than a pre-defined threshold, end the algorithm, else, go to step 3.
```

Step 1, initialization (get the global cluster centers). Regard all of the patterns that are labeled by users as global cluster centers. These centers will not be changed during iterations.

Step 2, label propagation. To reasonably spread the type of labeled patterns in the local area, we classify the patterns which fall in the threshold of cluster center as the same type of cluster center and set their corresponding certainty to the ratio of their distance to the threshold. If a pattern is in the thresholds of multiple cluster centers, its certainty is calculated according to its distances between these centers.

Step 3, update the membership matrix. Add local cluster centers and calculate membership matrix according to equ. 2. Because the number of labeled patterns is small at the beginning, we regard other patterns within the threshold of the current pattern also as cluster centers when calculating the membership of each pattern, so that the speed of the propagation of labels can be limited.

Step 4, normalization (calculate the type certainty). As each cluster center only has three types: normal, abnormal and uncertain, so a pattern's membership of belonging to a cluster center can be added to the type of that cluster center. The type certainty is finally normalized to represent the likelihood of this pattern that belongs to each type.

Step 5, comparison. Calculate the change of previous membership matrix and the new one. If the change is less than a predefined threshold, stop the algorithm, otherwise return to the third step.

With the increasing number of user-labeled patterns and the adjustment of patterns types, the result of the algorithm tends to be stable. If the certainty of a pattern being normal/abnormal is larger than a threshold, our system will classify it as normal/abnormal.

4.4 Influence calculation

As we know, AL makes candidate selection based on some strategies, which include uncertainty sampling, error reduction schemes, relevance-based selection and purely data-centered strategies (Bernard et al. 2018a).

Although our approach allows users to make independent candidate selection, suggestions and hints can be provided to users in the same way. Users are free to decide whether to adopt a suggestion according to their experience and the information such as instance distribution.

Because that our algorithm in Sect. 4.3 spreads the labels outward, we suppose that users tend to select the candidate instance that can influence other instances to the maximum extent. This may be considered as a density-based selection strategy, which is helpful in the case when no labels are available.

The influence degree of each instance is calculated initially and recalculated after user labeling. The calculation of the influence degree of the ith instance is as follows. Calculate the membership matrix after labeling the ith instance once (without iteration). Use the Euclidean distance between this matrix and the origin membership matrix as its influence degree. The time complexity of calculating the influence degrees of all the instances is $\{O(n^2)\}$, where n is the number of instances. The efficiency can be increased with parallel computing.

In the visualization module, we map the influence degree of a behavior pattern to the radius of the circle. A larger circle means that it has greater influence, and its label will spread faster (more nearby patterns will be affected) if labeling this pattern.

4.5 Incremental learning

With new data added, the AE and SOINN will be updated incrementally.

When the amount of incoming data reaches the batch number, the deep AE will be updated. The update process is the same as training.

Then, we update the SOINN. The update process is also the same as training: find the wining nodes and update their parameters such as weight and threshold. Then, recalculate the certainty. If there is no new node generated, iterate the membership matrix directly. Otherwise, set the type of new node to uncertain and its certainty to 1 and also calculate the membership matrix iteratively. Because each node can be viewed as a pattern and the number of patterns is usually limited, so the number of nodes will not be too large. Therefore, the incremental training can be finished in seconds. Finally, the classification algorithm is applied once again to adapt to the new data.

5 Interactive visualization interface

This section describes the visualization interface, which can be viewed as a bridge between analysts and the model by labeling patterns manually, so that the detection rate can be increased. We designed a visualization interface to help analysts to determine the pattern that needs to be labeled next and judge whether a pattern is abnormal or not.

5.1 SOINN topology view

For the central part of our approach: smart labeling, a main view is needed to display the current classification result and the pattern distribution. This can assist users in selecting candidate pattern for labeling and learn how their last labeling affect the classification results.

Figure 1a is the topological structure of SOINN nodes (network behavior patterns). As each node represents a ten-dimensional vector in our system, to visualize them while maintaining the high dimensional distances among the nodes, we use t-SNE (T-distributed Stochastic Neighbor Embedding) (Lvd and Hinton 2008) to reduce the vector to two dimensions.

The color of the nodes encodes the predicted labels. Orange means it is a normal pattern, and blue means abnormal. Nodes with black circles in the center mean that these nodes have been labeled manually by the user, which can be thought as ground truth. The brightness of color implies the node's certainty of belonging to a current type. If a node has very high certainty, its color will be light. The size stands for the influence of the node. Labeling a node with larger influence degree will affect more nodes nearby.

It can be seen that the nodes are gathered together, forming clusters. Nodes which are close in two - dimensional layout may also close in high dimension, representing similar network behavior patterns. Because of the distortion caused by dimension reduction, nodes that are close in high dimension are also connected with lines to avoid misleading.

Such a layout can assist in choosing the patterns to be labeled. Users can choose several patterns from each cluster to verify their types first so that the whole data can be roughly classified. In the same cluster, users can choose bigger patterns to label first. Labeling bigger patterns first can improve efficiency. After the rough labeling in the first round, users can then further choose some patterns which have higher uncertainty to verify and label again. The labeling is realized by right-clicking the pattern and choosing a type, as shown in Fig. 3a.

If the type of a pattern is changed during labeling, we suppose that it should be further labeled. We use red strokes to highlight them, as shown in Fig. 5a. Usually, this can arise when patterns are near the boundary of normal and abnormal clusters.

To analyze how the labeled patterns affect other patterns, we also provide the functions of viewing history types and checking the current impact factors of labeled patterns. When clicking a node, the history of how its type changed during labeling will be displayed above the node, as shown in Fig. 5b. Each cell represents for a historical type. The labeled pattern which caused the change of type will be highlighted with mouse hovering on the cell.

5.2 Matrix view

In order to further help users to choose the patterns to be labeled in the next step, we need another view to compare the similarity among different patterns and the feature vectors of time slices. As this is the display of an attribute among two different types of entities, matrix view is appropriate here.

First, there are also certainty values among the feature vectors of one-min time slices and behavior patterns. If the feature vector of a time slice only has one matched pattern, assume the Euclidean distance between them is d, and the threshold of this pattern is $thre$. Then, the feature vector of the time slice's certainty of being related to this pattern can be calculated by: $thre/(d + thre)$. If there are two matched patterns: s_1 and s_2, assume the distances to them are d_1 and d_2. Then, the certainty of the feature vector of this time slice to s_1 is: $d_2/(d_1 + d_2)$; the certainty to s_2 is: $d_1/(d_1 + d_2)$.

The matrix view is shown in Fig. 1b. Each row is a pattern, and each column is a one-min time slice. The cell represents the time slice's certainty of being related to the pattern. The higher the certainty is, the lighter the color is. The black color means that there is no relationship between the pattern and the time slice.

After clicking a pattern, the first row of the matrix will be changed to that pattern, and the shown time slices will be all of the times that have relation to this pattern. Others patterns which are related to these time slices will also be drawn as the following rows. These patterns can be considered to have high similarity to the selected pattern. When clicking a time slice in the matrix view, all the other views will show information during that time.

Through this view, users can know which times are most related to the selected pattern, so that they can select the time to verify the network status further. Moreover, this view provides similar patterns to help labeling. When a user finds an abnormal pattern, he can use this function to find more similar abnormal patterns.

Above the matrix is the global time distribution of the shown time slices. By hovering on the time slices, the corresponding bar at the time distribution will be highlighted. In this way, users can discover the relation between time slices and the possible occurrence of events.

5.3 Auxiliary views

Then, we describe some auxiliary views which can help users to judge whether a node is normal or abnormal.

- *Network structure* Network topological structure is beneficial to understanding the network situation. The force-directed layout in Fig. 1e shows the network topological structure during the selected time period. Each oblong node represents an IP. The height and width of a node indicate its inbound and outbound traffic. Servers are usually displayed as spindly rectangles while clients are usually wide and flattened rectangles.
- *Timeline overview* In order to observe the traffic flow from multiple scales, we divide timelines into three parts: days, hours and minutes (from left to right), as shown in Fig. 1f. Timelines use the form of a stacked histogram to show the percentage of traffic from different protocols/ports. Users can click the radios on the left side to decide whether to split the data on the basis of protocols or ports.

- *Statistics information* The boxplot/scatter in Fig. 1d shows the distributions of the destination port, number of records and network traffic of different IPs (each node represents an IP). These attributes are useful for discovering abnormal IPs from a different point of view. For instance, when the *x*-axis is set to destination ports and the *y*-axis is set to the number of records, it shows the distribution of request times at different destination ports. Thus, IPs that initiated too many requests can be filtered. Users can switch to a scatter plot to observe detailed information.
- *Trend charts* The first line chart in Fig. 1c shows the entropy curves of the source IP, destination IP and destination port. Because the entropies always change significantly when attacks occur, displaying information entropy is beneficial to anomaly detection. The next two line charts show the traffic-time curve of two main IPs. The traffic curves of the IPs selected in the topological view will also be shown in this part.

6 Case study

We use the Challenge 3 of VAST Challenge 2013 which is a popular dataset for network security visual analysis (Shi et al. 2018, 2015; Zhao et al. 2014) to verify the effectiveness of our system.

The data in the first week include: Network description, NetFlow data and Network health and status data (Big Brother data). Organizationally, Big Marketing consists of three different branches, each with around 400 employees and its Web servers. The data volume of the first-week data is about 5G. Given universality, we only use the NetFlow data in the following analysis.

6.1 Normal patterns

Figure 3a shows the mapping of network behavior patterns in two dimensions. Initially, all the patterns are unlabeled and unclassified. We first select some patterns with larger radius from different clusters to verify their types. To explain, we number them with cluster id.

We choose the pattern in cluster 1 to label first because this cluster has a higher influence degree. Click this pattern and choose the most related time slice (the lightest ones) in the matrix view. There is no abnormality found in auxiliary views, so we judge that this pattern is a normal pattern. We label it as normal, as shown in Fig. 3a.

After this labeling, the result is shown in Fig. 3b. The labeled pattern has an impact on neighboring patterns. Patterns close to it are also classified as normal. It can also be observed that, after labeling, the

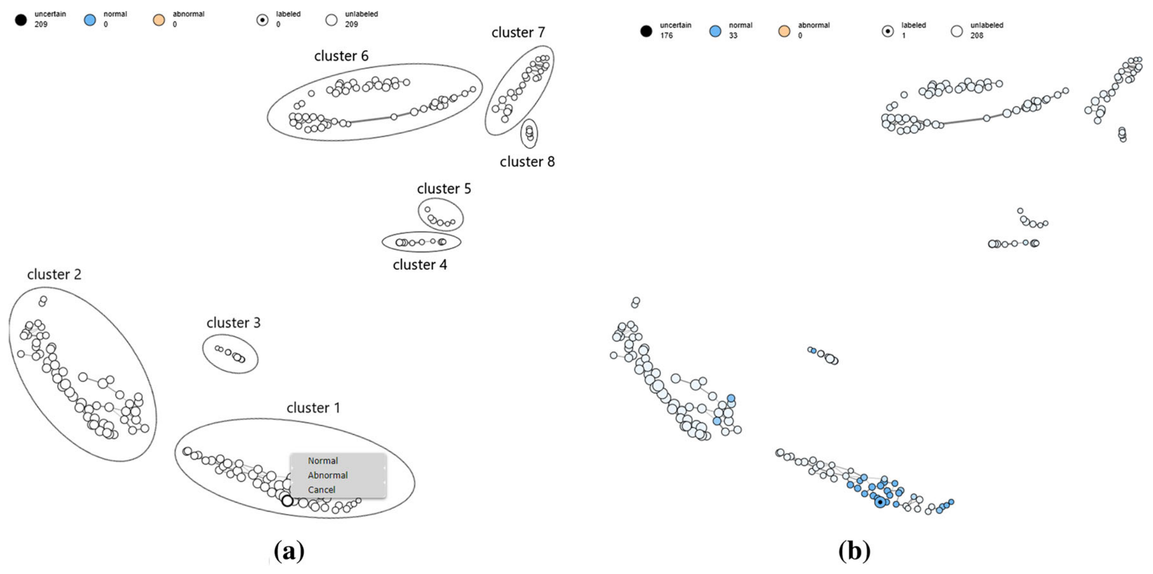


Fig. 3 The process of labeling. **a** The distribution of network behavior patterns. Initially, all of the patterns are unlabeled and unclassified. **b** The result after labeling a normal pattern

influence degrees of the patterns around the labeled one are decreased (smaller radius), and the influence degree of the patterns in cluster 2 is increased. So we can continue to label the patterns in cluster 2.

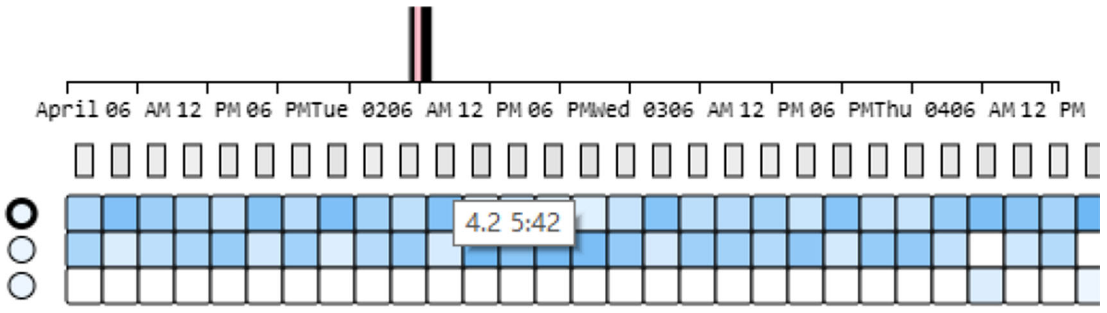
6.2 DDoS attacks

Continue to label the patterns with large radius, we find an anomaly when labeling a pattern in cluster 5 (Fig. 5a).

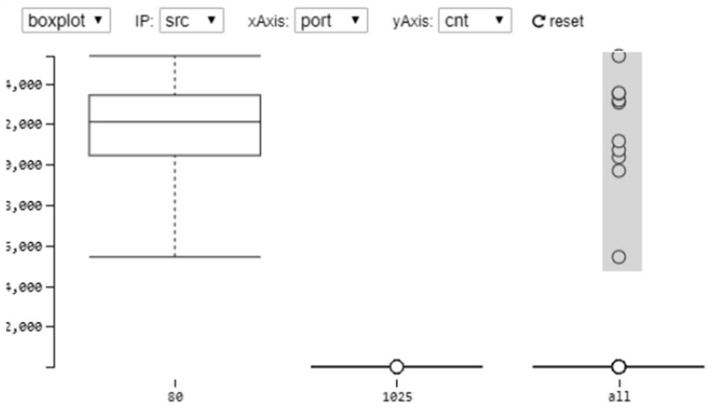
The matrix view is shown in Fig. 4a, and we click the time slice with the highest certainty (the lightest bar): 5:42, April 2nd, other views then show corresponding information. Fig. 4b shows that several source IPs connect to port 80 with high traffic. Selecting these IPs to highlight them in other views, Fig. 4c shows that these IPs visit the server 172.30.0.4 simultaneously. From the thickness of links, we can judge that the traffic of these connections is much larger than that of normal connections, which is a DDoS attack, so we right-click the pattern and label it as abnormal.

After labeling this abnormal pattern, the result is shown in Fig. 5a. It can be seen that the types of the patterns in cluster 4 are mixed. Some patterns are more likely to be normal, while others seen to be abnormal. The labeling also cause the type of a pattern (with red stroke) in cluster 4 to be changed from normal to abnormal, as shown in Fig. 5b. So we need to further label the patterns in cluster 4.

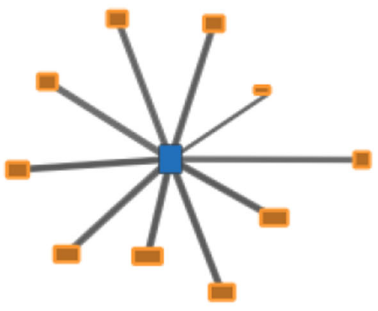
By the same means, we find that this pattern also stands for a DDoS attack which occurred at 9:38, April 3rd. The only difference is that multiple servers are attacked. Therefore, we labeled it as abnormal. After labeling, the final result is shown in Fig. 5c. We also find that the patterns in the matrix views of this cluster all have similar behaviors. This means that our system classified most of the similar anomalies correctly and the matrix view also captured similar patterns correctly.


April 06 AM 12 PM 06 PM Tue 02 06 AM 12 PM 06 PM Wed 03 06 AM 12 PM 06 PM Thu 04 06 AM 12 PM
4.2 5:42

(a)



(b)



(c)

Fig. 4 The discovery of an abnormal pattern. **a** Time slices that are related to the selected pattern. Choose the most related one to judge whether there are anomalies or not. **b** Select IPs that had large traffic. **c** The main topological structure of DDoS

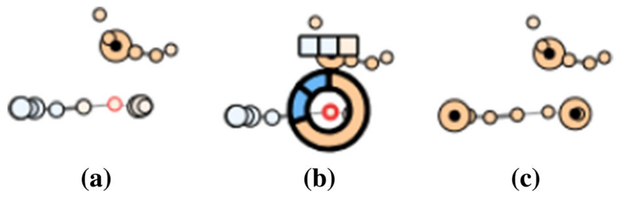


Fig. 5 The adjustment process after labeling the first abnormal pattern. **a** After labeling the first abnormal pattern, we find that normal and abnormal patterns are mixed. **b** Some patterns are changed from normal to abnormal. **c** The result at the end of the adjustment

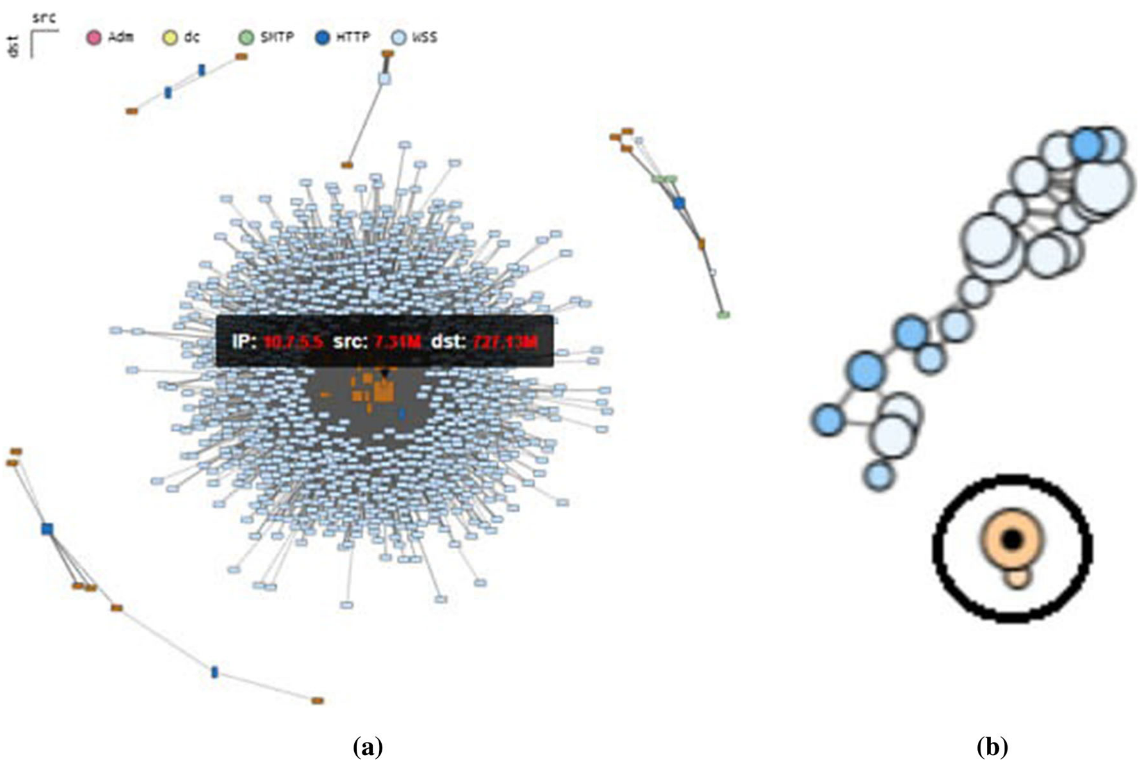


Fig. 6 The redirection attack. **a** The topology shows that IP 10.7.5.5 has a lot of communication, **b** the labeling result

6.3 Redirection attack

We then continue to label the patterns in cluster 8. The timelines show that these patterns are related to a large number of time slices among 7:05, April 4th, to 8:51. As shown in Fig. 6a, the IP topology shows that the external IP 10.7.5.5 communicate with many internal workstations. These workstations initiate connections, and then, IP 10.7.5.5 transfer files back with high traffic. Note that this IP participated in the DDoS attacks discovered before, which was an attacker. Then, we further check the relevant information of the several servers. It can be found that the duration time and package number of 172.20.0.4 are obviously reduced during this time. This anomaly is caused by the malware that was inserted to 172.20.0.4, and all the visitors of this server are redirected to 10.7.5.5. Figure 6b shows the labeling result.

6.4 IP scan

So far, we have demonstrated how to select patterns according to the distribution and suggestions and then label them. Another way to label patterns without the use these information is by monitoring other views and labeling the related patterns of time. Tracing detected attackers is also a way to discover subtle anomalies.

For instance, tracing the attacker 10.6.6.6, we find the anomalies at 13:22, April 2nd. As shown in Fig. 7c, we switch the boxplot view to the scatter view and set the x-axis to the number of connections and

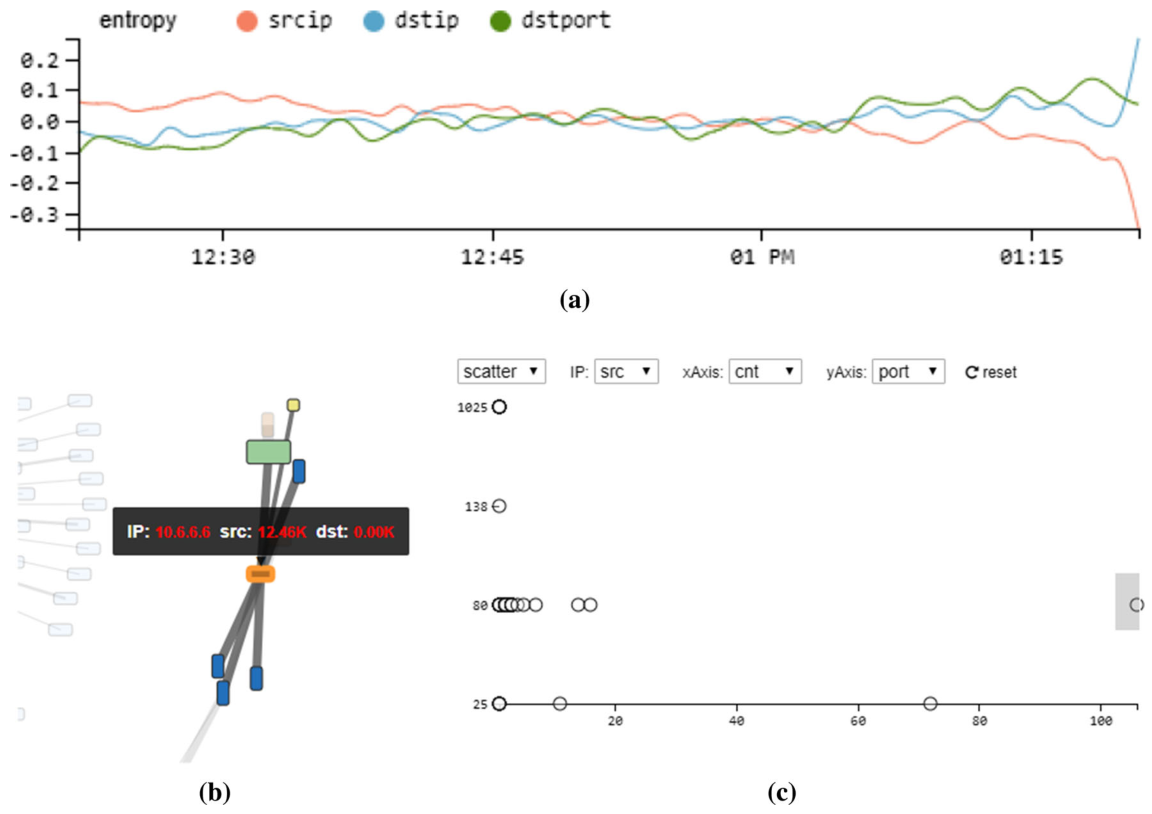


Fig. 7 An IP scan. **a** The sudden change of entropy, **b** the topological structure of scan, **c** scatter plot shows that 10.6.6.6 accessed port 80 of these servers more than a hundred times

y-axis to destination ports. We select the outlier that had a substantially large number of connections and then turn to the network topology to check the corresponding highlighted IPs. As shown in Fig. 7b, 10.6.6.6 accessed port 80 of several servers more than a hundred times. The entropy curves (Fig. 7a) show that the entropy of the source IP decreased suddenly, while the entropy of the destination IP increased, which is a typical IP scan. Choose this time slice, and the related pattern in the main view will be highlighted.

6.5 Data exfiltration

Switch the timelines to the stacked port mode. From Fig. 8, we find that there are two abnormal FTP traffic records on April 6th and April 7th. Other auxiliary views show that this was caused by the attacker 10.7.5.5 who transferred a large amount of data. Combined with the redirection events we discovered in Sect. 6.3, we guess that this attacker infected some hosts by redirection to get the access to FTP transmission and then leaked the data of this company.

Thus, it can be seen that our system can discover and label anomalies such as DDoS attacks, IP scan, redirection attacks and data exfiltration.

After being labeled about 20 times, all of the patterns have their most possible types (Fig. 9). Patterns with lower certainty can be chosen repeatedly to optimize the system. Therefore, in real-world scenes where new data will occur, the system can better detect anomalies automatically. Users can also verify the type of new patterns regularly to improve the accuracy of the system classification further.

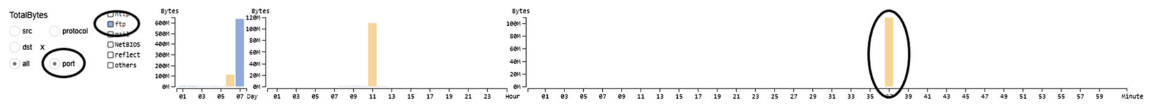


Fig. 8 An FTP anomaly with 100M traffic transmission

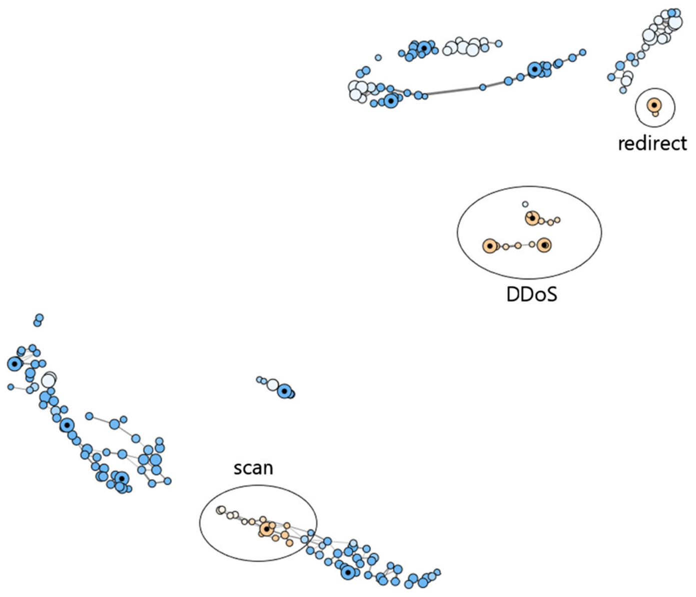


Fig. 9 The final result. Different types of attacks are labeled

7 Experiments

In this section, we compare the selection strategy of instances for labeling of our approach with other strategies. We designed three contrast experiments to verify that our system can improve the efficiency and accuracy of labeling. All the patterns default to normal at the beginning.

Still use the NetFlow data in Challenge 3 of VAST Challenge 2013. The dataset is divided into normal and abnormal manually. We use true positive rate (TPR) and false positive rate (FPR) to evaluate the capability of our system because these are the two aspects that we are most concerned of and want to improve.

- *Experiment 1 (E1)* Visual interactive selection of our approach. Select and label 30 patterns according to the influence suggestion and the distribution of patterns by users themselves. This process is similar to the cases demonstrated in Sect. 6.
- *Experiment 2 (E2)* Random selection. Randomly select 30 patterns and label them with gold labels.
- *Experiment 3 (E3)* AL strategies. Use AL strategies to select 30 candidate patterns and make labeling with gold labels. We choose three AL query strategies and two classifiers. The three query strategies are: Smallest Margin (Scheffer et al. 2001), Entropy-Based Sampling (Settles and Craven 2008) and Least Significant Confidence (Culotta and Mccallum 2005). The two classifiers are: SVM and Logistic Regression.

We asked ten computer science students (2 graduate student, 8 undergraduate student) to do the E1. These users all have basic knowledge of the area of network security and can correctly judge the network status according to the auxiliary views. They were introduced to the goals of the experiment at the beginning. Then, we used a simple demo to show the usage of the system and briefly explained how it worked. Finally, they were asked to complete the E1. All participants finished the experiment in 30 min. The results are shown in Figs. 10 and 11.

Compare the results in Fig. 10a to test the accuracy of our approach. It can be seen that the TPR of E1 is the highest. This implies that the pattern distribution and the influence selection help users a lot in deciding the patterns to be labeled. It also implies that the detection rate can be increased substantially with manual intervention and interaction. E2 has the lowest TPR because the patterns to be labeled are decided randomly, and the probability that the abnormal patterns are selected is relatively low. Therefore, both the TPR and the FPR of E2 are low. Although the FPR of E1 is larger than E2 and E3 in Fig. 10b, it is still acceptable and can also be decreased with more labeling times.

Figure 11 shows how the TPR and FPR change with the increase in labeling times in E1 and E3. The change trend of E2 is ignored because it has large randomness, and the average result is not meaningful.

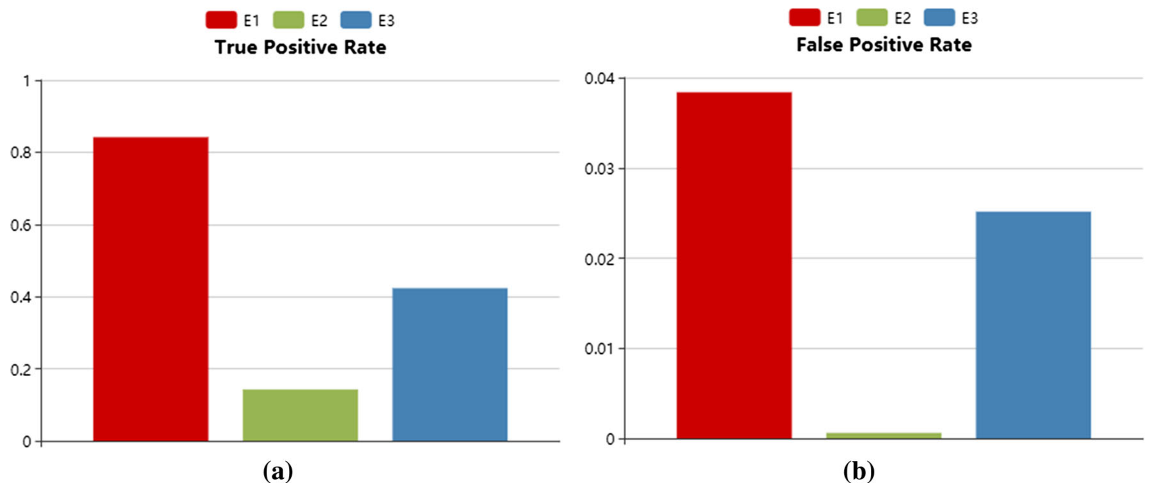


Fig. 10 The comparison of the average final results of the experiments

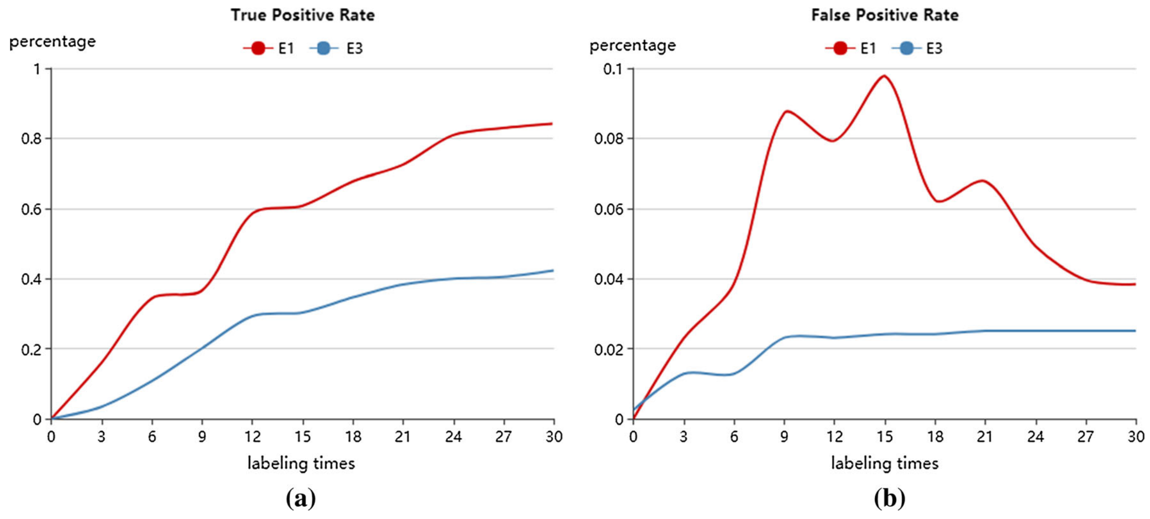


Fig. 11 The average TPR and FPR of E1 and E3 with the increase in labeling times. **a** True positive rate and **b** false positive rate

It can be observed that, with the increase in the number of labeled patterns, the detection rate is improved. The average final TPR of E1 reaches 84.71%. Although the FPR of E1 fluctuated during the labeling process, it also tended to be stable, and the average final FPR of E1 is 3.83%. If users continue labeling, we can assume that the performance will be further improved. (The FPR can be reduced to 1.74% with 10 more labeling times on average.)

Comparing the TPR of E1 and E3 in Fig. 11a, it shows that E1 (our approach) precedes E3 (AL strategies). Besides, the detection rate of our approach rises faster than E3, which means that our approach can obtain higher TPR with less labeling times.

Comparing the FPR of E1 and E3 in Fig. 11b, E1 (our approach) has larger fluctuation during labeling (especially when the labeling times is small), first increased and then decreased. E3 (AL strategies) is relatively stable, and the average final result of FPR is 2.51%.

8 Conclusion and future work

In this paper, we propose an interactive labeling platform for anomaly detection which combines the semi-supervised learning methods, active learning and visualization techniques to give suggestions for candidate selection, help users understand the model and take full advantage of the unlabeled data, so that the machine learning methods could be further utilized in real-world cybersecurity situations. When the user labels a pattern, the algorithm will classify other patterns with confidence or probability and calculate the influence degrees. Besides, on account of the incremental learning ability of our method, our work can be applied to situations with dynamic data.

However, the label work should be done by the domain experts or be done by many people to make sure it is of high confidence. Also, in our algorithm, we suppose that the confidences of manual labels are 1. So the people who do the labeling work should have certain cyber security knowledge and understand how our model works. Additional research could be done to take the confidence of the manual labels into consideration. What's more, the labeling granularity can be further refined to specific attack types.

Acknowledgements Supported by National Key Research and Development Program of China (Grant Nos. 2016QY02D0304, 2017YFB0701900), National Nature Science Foundation of China (Grant No. 61100053), and Key Laboratory of Machine Perception in Peking University (Grant No. K-2019-09).

References

Ahmed M, Mahmood AN, Hu J (2016) A survey of network anomaly detection techniques. J Netw Comput Appl 60:19–31

Aljawarneh S, Aldwairi M, Yassein MB (2018) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J Comput Sci 25:152–160

Almgren M, Jonsson E (2004) Using active learning in intrusion detection. In: Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. IEEE, pp 88–98

Bernard J, Sessler D, Bannach A, May T, Kohlhammer J (2015) A visual active learning system for the assessment of patient well-being in prostate cancer research. In: Proceedings of the 2015 workshop on visual analytics in healthcare. ACM, p 1

Bernard J, Hutter M, Zeppelzauer M, Fellner D, Sedlmair M (2018a) Comparing visual-interactive labeling with active learning: an experimental study. IEEE Trans Vis Comput Graph 24(1):298–308

Bernard J, Zeppelzauer M, Sedlmair M, Aigner W (2018b) VIAL: a unified process for visual interactive labeling. Vis Comput 34(9):1189–1207

Bhuyan MH, Bhattacharyya DK, Kalita JK (2014) Network anomaly detection: methods, systems and tools. IEEE Commun Surv Tutor 16(1):303–336

Bruns-Smith D, Baskaran MM, Ezick J, Henretty T, Lethin R (2016) Cyber security through multidimensional data decompositions. In: Cybersecurity symposium (CYBERSEC), 2016. IEEE, pp 59–67

Buczak AL, Guven E (2016) A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun Surv Tutor 18(2):1153–1176

Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv (CSUR) 41(3):15

Chen W, Kong F, Mei F, Yuan G, Li B (2017) A novel unsupervised anomaly detection approach for intrusion detection system. In: Big data security on cloud (BigDataSecurity), IEEE international conference on high performance and smart computing (HPSC), and IEEE international conference on intelligent data and security (IDS). IEEE, pp 69–73

Culotta A, Mccallum A (2005) Reducing labeling effort for structured prediction tasks. In: National conference on artificial intelligence, pp 746–751

Dunn JC (1973) A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. J Cybern 3(3):32–57

Fischer F, Mansmann F, Keim DA, Pietzko S, Waldvogel M (2008) Large-scale network monitoring for visual analysis of attacks. In: International workshop on visualization for computer security. Springer, Berlin, pp 111–118

Fritzke B (1995) A growing neural gas network learns topologies. In: International conference on neural information processing systems, pp 625–632

Furao S, Hasegawa O (2006) An incremental network for on-line unsupervised classification and topology learning. Neural Netw 19(1):90–106

Görnitz N, Kloft M, Rieck K, Brefeld U (2009) Active learning for network intrusion detection. In: Proceedings of the 2nd ACM workshop on security and artificial intelligence. ACM, New York, pp 47–54

Heimerl F, Koch S, Bosch H, Ertl T (2012) Visual classifier training for text document retrieval. IEEE Trans Vis Comput Graph 12:2839–2848

Höferlin B, Netzel R, Höferlin M, Weiskopf D, Heidemann G (2012) Inter-active learning of ad-hoc classifiers for video visual analytics. In: Visual analytics science and technology (VAST). IEEE, pp 23–32

Kohonen T (1990) The self-organizing map. Proc IEEE 78(9):1464–1480

Lin H, Gao S, Gotz D, Du F, He J, Cao N (2018) Rclens: interactive rare category exploration and identification. IEEE Trans Vis Comput Graph 24(7):2223–2237

Livnat Y, Agutter J, Moon S, Foresti S (2005) Visual correlation for situational awareness. In: IEEE Symposium on information visualization, 2005. INFOVIS 2005. IEEE, pp 95–102

Lvd M, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9(Nov):2579–2605

Mansmann F, Keim DA, North SC, Rexroad B, Sheleheda D (2007) Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. IEEE Trans Vis Comput Graph 13(6):1105–1112

McPherson J, Ma K-L, Krystosk P, Bartoletti T, Christensen M (2004) Portvis: a tool for port-based detection of security events. In:Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM, New York, pp 73–81

Mukherjee B, Heberlein LT, Levitt KN (1994) Network intrusion detection. IEEE Netw 8(3):26–41

Scheffer T, Decomain C, Wrobel S (2001) Active hidden Markov models for information extraction. In: International symposium on intelligent data analysis. Springer, Berlin, pp 309–318

Settles B (2009) Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences

Settles B, Craven M (2008) An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 1070–1079

Shi R, Yang M, Zhao Y, Zhou F, Huang W, Zhang S (2015) A matrix-based visualization system for network traffic forensics. IEEE Syst J 10(4):1350–1360

Shi Y, Zhao Y, Zhou F, Shi R, Zhang Y (2018) A novel radial visualization of intrusion detection alerts. IEEE Comput Graph Appl 38(6):83–95

Shiravi H, Shiravi A, Ghorbani AA (2012) A survey of visualization systems for network security. IEEE Trans Vis Comput Graph 18(8):1313–1329

Sommer R, Paxson V (2010) Outside the closed world: on using machine learning for network intrusion detection. In: 2010 IEEE symposium on security and privacy (SP). IEEE, pp 305–316

Sudo A, Sato A, Hasegawa O (2009) Associative memory for online learning in noisy environments using self-organizing incremental neural network. IEEE Trans Neural Netw 20(6):964–972

Velea R, Ciobanu C, Gurzau F, Patriciu V-V (2017) Feature extraction and visualization for network PCAPNG traces. In: 2017 21st international conference on control systems and computer science (CSCS). IEEE, pp 311–316

Yousefi-Azar M, Varadharajan V, Hamey L, Tupakula U (2017) Autoencoder-based feature learning for cyber security applications. In: 2017 international joint conference on neural networks (IJCNN). IEEE, pp 3854–3861

Zhang S, Fung C, Huang S, Luan Z, Qian D (2017) Psom: periodic self-organizing maps for unsupervised anomaly detection in periodic time series. In: 2017 IEEE/ACM 25th international symposium on quality of service (IWQoS). IEEE, pp 1–6

Zhao Y, Liang X, Fan X, Wang Y, Yang M, Zhou F (2014) MVSec: multi-perspective and deductive visual analytics on heterogeneous network security data. J Vis 17(3):181–196

Zhao Y, Luo F, Chen M, Wang Y, Xia J, Zhou F, Wang Y, Chen Y, Chen W (2018) Evaluating multi-dimensional visualizations for understanding fuzzy clusters. IEEE Trans Vis Comput Graph 25(1):12–21