TVi: A Visual Querying System for Network Monitoring and Anomaly Detection

Alberto Boschetti
University of Brescia
NTW group
Brescia, Italy
alberto.boschetti@ing.unibs.it

Chris Muelder
University of California Davis
VIDI Lab
Davis, CA, USA
cwmuelder@ucdavis.edu

Luca Salgarelli*
University of Brescia
NTW group
Brescia, Italy
luca.salgarelli@ing.unibs.it

Kwan-Liu Ma
University of California Davis
VIDI Lab
Davis, CA, USA
ma@cs.ucdavis.edu

ABSTRACT

Monitoring, anomaly detection and forensics are essential tasks that must be carried out routinely for every computer network. The sheer volume of data generated by conventional anomaly detection tools such as Snort often makes it difficult to explain the nature of an attack and track down its source. In this paper we present TVi, a tool that combines multiple visual representations of network traces carefully designed and tightly coupled to support different levels of visual-based querying and reasoning required for making sense of complex traffic data. TVi allows analysts to visualize data starting at a high level, providing information related to the entire network, and easily move all the way down to a very low level, providing detailed information about selected hosts, anomalies and attack paths. We designed TVi with scalability and extensibility in mind: its DBMS foundations make it scalable with virtually no limitations, and other state-of-the-art IDS, like Snort or Bro, can be easily integrated in our tool. We demonstrate with two case studies, a synthetic dataset (DARPA 1999) and a real one (University of Brescia, UniBS, 2009), how TVi can enhance a network administrator's ability to reveal hidden patterns in network traces and link their key information so as to easily reveal details that by merely observing Snort's output would go unnoticed. We make TVi's source code available to the community under an Open Source license.

Categories and Subject Descriptors

C.2.3 [**Computer Communication Networks**]: Network Monitoring

*This work was supported in part by a grant from the Italian MIUR, under the PRIN project *IMPRESA*.

VizSec '11, July 20, 2011, Pittsburg, PA, USA

General Terms

Design, Experimentation

1. INTRODUCTION

Since the beginning of the Internet, network monitoring and anomaly detection have become two of the most crucial tasks of modern network management, for which many tools have been developed. The main goal of these tools is to provide an accurate evaluation of the network status over time. One of the most popular NIDS (Network Intrusion Detection System) is Snort [8]. By parsing the content of every packet, Snort looks for fingerprints of known attacks. Its output is text–based and analysing the resulting data can be daunting, especially for large–scale networks. For example, during the second week of the DARPA 1999 challenge dataset [27] the Snort output (running in text logging mode) is more than 18,000 entries long, and for the 30 hours dataset collected in 2009 of the Engineering building of the University of Brescia (UniBS 2009), the size grows to more than 200,000. An exhaustive analysis of such an amount of information cannot be done by hand, but becomes easier with the help of a visualization tool. Even so, the huge log file is not completely useful, because the logging method is per–packet, i.e., for one IP scan many entries are created.

Many visualization tools have been proposed in recent years [29]. Every tool has a well–defined goal, with some advantages and some disadvantages. The most common limitations are related to the scalability of the tool, the covered features, and its dynamic properties. In fact, some well-know tools can only manage small–size networks, some can detect only a few categories of anomalies, and some others are able to visualize only a few particular aspects of the network, e.g., throughput, rates or protocols. Another problem comes from the input data: by using a logfile as input of the process, the system is hardly scalable and some operations related to the data filtering are very difficult and time–consuming. For instance, if we only want to see the connections on TCP port 80 with a visualization system based on logfiles, the process can be done in one of two ways. In the first one, the program would parse the entire file and keep only the interesting lines; the second method is to load the entire file in memory and then query it. In the first case, we

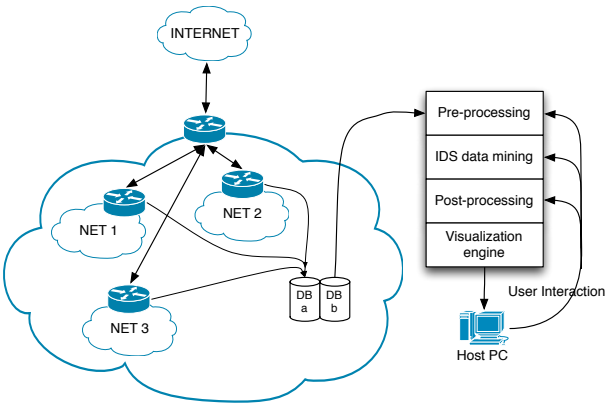


Figure 1: TVi deployment: an example

lose interaction, as for every query a complete scan of the file must be done, while in the second case the program is not scalable because of memory occupancy.

We designed TVi (Trace Visualizer) with the stated goal of solving these issues. Its main contributions to traffic visualization, which we will describe in the rest of the paper, are the following.

Integration of visual representation and machine learning data-processing tools: Principal Component Analysis (PCA) transforms and analysis based on data entropy are at the base of the data which is visualized by the end user, allowing them to see only relevant data to a particular analysis goal, e.g., anomaly detection, as opposed to scanning huge logfiles.

Very high scalability: TVi uses relational databases to store its data. By leveraging the distributed features of modern DBMS, it can easily scale to cover a small LAN with a single point of data capture to operational backbones with tens of traffic probes. Figure 1 shows an example deployment scenario.

Multiple, optimized visualization tools: TVi's user interface combines many 2D visualization methods (multi–view): histograms, timelines, graphs, geo–clustered graphs and matrices. All the views are linked together; therefore, the same feature can be shown in different ways in order to investigate different aspects of an anomaly. Its integration with other NIDS such as Snort makes it then easier to pinpoint the sources of anomalies.

We make TVi's code available to the community under an Open Source license at [10].

The rest of this paper is organized as follows: Section 2 describes related works. Section 3 and Section 4 explain the architecture of TVi and its main implementation criteria. Section 5 details a step-by-step use of the system through two case studies. We discuss the preliminary results we obtained running TVi, and advocate for the community to test it in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORK

Many solutions have been proposed in the literature, and an ample collection of them is described in [29]. Most of the mature tools, such as NVisionIP [25], Rumint [14], TimeSearcher [13], and TNV [9] are not scalable and use a non query–able input format; they load all the input data and process it internally. By using them it is also impossible to update the data, because the only way to do so is to clear the memory, reload and reprocess the input. Some tools, such as Rumint, TimeSearcher and TNV visualize many aspects of the data in the same window. They split the window in multiple sub–windows; each of them is specialized in a graphical representation. They can visualize in the subwindows either high–level (aggregated statistics) or low–level information of the network (such as IP connections), but the partition makes difficult to analyse and understand networks composed by hundreds of nodes or more. Some programs narrow their focus to just one feature of the network. PortVis [31], for instance, only analyses traffic patterns with respect to aggregate TCP port usage.

In order to present more parameters through the visualization screen, The Spinning Cube of Potential Doom [26] and [39] use a different approach, proposing a 3D visualization tool. Each axis represents a different feature of the network; therefore, a point in this three–dimensional space is the map of a connection. This type of visualization is pretty hard to understand, even in presence of a few nodes because of typical problems with 3D visualization, such as occlusion.

More similar to our contribution, FloVis [38] with the plugin [16] allows the creation of many different visualization environments for observing and monitoring the network from many points of view. However, this tool works only on TCP traffic at the flow level, and it requires an accurate tuning of the parameters and the hierarchy in the observed network. In contrast, TVi does not require any a–priori knowledge of the network architecture, it works with multiple protocols and it can analyse the network at different levels (raw traces, flow level, node level, geolocalization level, etc).

Finally, very few visualization tools integrate machine learning algorithms for anomaly detection; in most cases they rely solely on the users perception, as in [18]. Here, the authors propose a visual system for inspecting the network, by using several intertwined visualization tools that operate on an SQL database for storage. This makes the system scalable and reliable even with large datasets. Even though the visual system is complete, there is no automatic or even semi-automatic mechanism to alert the user about potential anomalies. TVi instead combines in a stand alone application an automatic anomaly detection system tightly coupled with a network visualizer.

3. TVI ARCHITECTURE: ANOMALY DETECTION

3.1 Feature Extraction

Most state-of-art algorithms for network anomaly detection capture many features of packet traces. All of them are combined and they serve as input to a machine learning engine, which processes the data to produce details about anomalies.

Patcha and Park in [34] provide an overview of the different methods used in the literature for discovering anomalous behavior in network traces. Techniques can be based on: statistical, machine learning (system call, Bayesian networks, principal component analysis, Markov models), data–mining (identification/classification, fuzzy logic, genetic, neural associative) and hybrid methods. Principal Component

Analysis (PCA) is one of the most widely used algorithms to make complex datasets more manageable. Through dimensionality reduction, PCA reduces both the time and complexity needed to process the data. Additionally, entropy vectors of the input features (a technique derived from information theory) are used in conjunction to space reduction: [33] and [24] show how entropy is a good metric for anomaly detection: it is computationally scalable, pretty accurate, and it can be efficiently used even in presence of sub-sampled network data, for example using opportunistic sampling as shown in [12].

PCA applied to entropy vectors is at the base of TVi's data processing engine, which we describe in the following.

3.1.1 Entropy

Let X be a discrete random variable representing a feature (in our case a network feature such as a port number, an IP address, or an application protocol) and x_i, with $i \in [1, N]$ the possible values of the feature X during the observations. The entropy of the random variable X, denoted as $H(X)$, is defined as:

$$H(X) = -\sum_{i=1}^{N} p(x_i) \cdot \log_2 p(x_i)$$

where $p(x_i)$ is the probability that the variable X takes the value x_i during the observation set; that is the ratio between the number of observations where X is x_i and the total number of observations.

Entropy is a measure of the data's randomness. If the data is concentrated in one point, i.e., all observations have the same value, its entropy is zero. Conversely, data spread out among many values will generate higher entropy. For example, during a portscan attack an intruder tries to connect to many ports (scan) of the target host (one IP address). In such situation, the histogram of the ports distribution of the destination host has a larger support than during regular operation, and the entropy of this histogram will be higher. During this attack, the histogram of the destination IP addresses has, instead, a high bin (the target IP of the portscan attack) and other normal–size bins. Therefore, its entropy will be quite low.

3.1.2 Features and database

The features used in the anomaly detection process are easy to retrieve and can be quickly obtained with standard protocols. The monitored features are related to the most common protocols (TCP, UDP and ICMP) and are flow-oriented (for TCP) or session-oriented (for ICMP and UDP), i.e. one observation for each flow/session[1]. The extracted feature vector dimensions are the source/destination IP address and the source/destination port (except for ICMP), because they are strictly correlated with network attacks (see for example [24] and [23]). We also save the number of packets and the quantity of bytes in the flow for use in the visualization matrix (see Section 4.4).

The features can be retrieved even at the border router, because the information is located in the packet headers at levels 3 and 4 of the TCP/IP stack. Cisco NetFlow [1] or SNMP are two standard protocols available on common routers which can be used for aggregating the needed features. The raw traces, e.g. Tcpdump output, can also be used for flow extraction. Once extracted, the flows are inserted in a DBMS table with the columns shown in Table 1.

[1]UDP session = UDP packets exchanged between two nodes without more than 1 minute of silence. ICMP session = ICMP packet and it eventual answer (e.g. ping echo and its reply).

Table 1: Raw features – Columns of the SQL table

1. timestamp, seconds since Epoch (*ts*);
2. source IP address (*sip*);
3. destination IP address (*dip*);
4. source port, if possible (*sport*);
5. destination port, if possible (*dport*);
6. protocol number (*proto*);
7. bytes into the flow (*byf*);
8. packets into the flow (*pkf*);

Table 2: Features whose entropy is fed as input to the PCA-based anomaly detector, computed every timeslice T.

1. Source IP - TCP protocol;
2. Source IP - UDP protocol;
3. Source IP - ICMP protocol;
4. Destination IP - TCP protocol;
5. Destination IP - UDP protocol;
6. Destination IP - ICMP protocol;
7. Source TCP port;
8. Source UDP port;
9. Destination TCP port;
10. Destination UDP port;

By using a DBMS, MySQL in our case, it turns out that data retrieval is fast enough to support near real-time visualization of the processed data. Even though general-purpose DBMS' transfer data slowly through the query interface, by opportunely tuning the parameter of the tables, i.e., creating indexes, keys and stored procedures, we were able to achieve a fast interaction between visual system and database. In order to give an idea of the timing, on an Apple Power Mac equipped with a 4-core Xeon @2.66GHz with 3GB of RAM and $MySQL$5.5, TVi can build the histogram of a feature in 115ms (without involving the cache), using a table composed by more than 13 million entries (UniBS09 dataset). The query in this case contains both `where` clauses and grouping.[2]

The scalability of the DBMS could be increased by using a distributed DBMS, such as a MySQL cluster, because in this case the workload would be split between the cluster nodes. Although this is still to be tested experimentally with TVi, it has been demonstrated [6] that MySQL clusters can effectively scale to the processing of virtually unlimited data size.

3.1.3 Anomaly Detection Algorithm

TVi uses a PCA-based technique for anomaly detection. The motivation is twofold: PCA is fast and achieves good results in terms of hit–rates [35, 33]. It is computationally efficient since it is based on matrix multiplication. Furthermore, PCA analysis could also be implemented in the DBMS itself, with a list of SQL queries [32], leveraging even further the design choice of basing our architecture on a database.

Starting from the database table, which contains all the

[2]SELECT timestamp, proto, sip, count(0) FROM <table> WHERE timestamp=<T> GROUP BY proto, sip

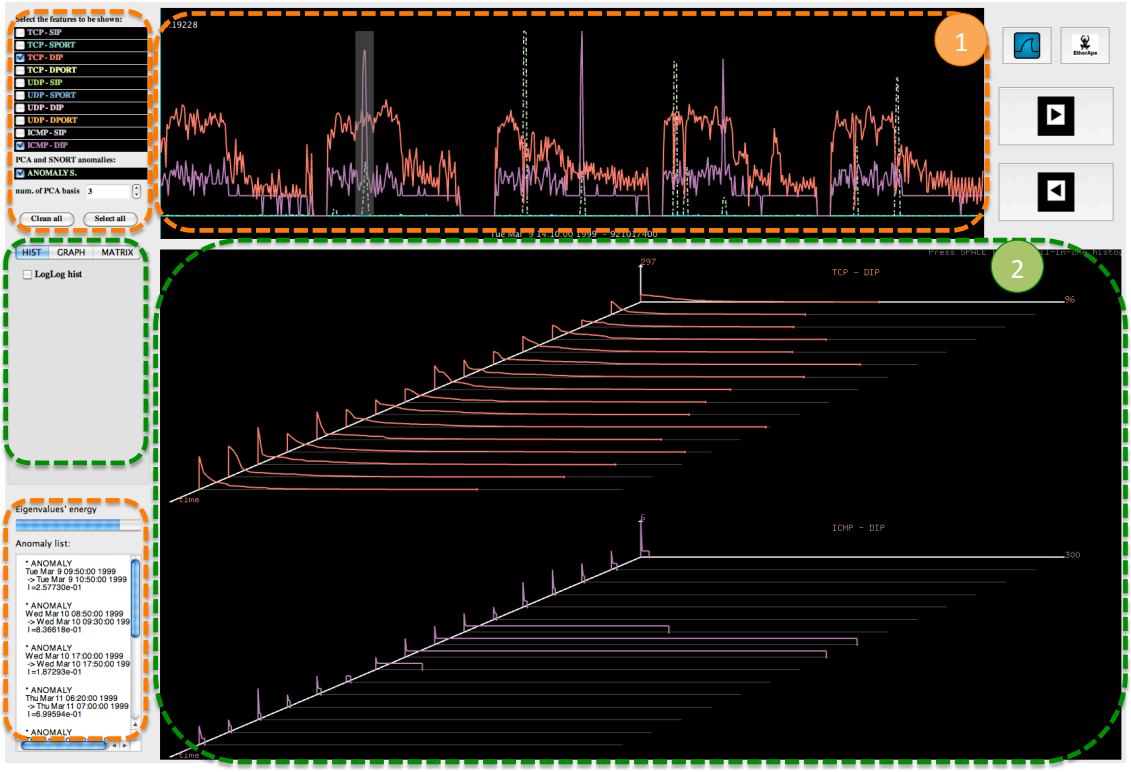


Figure 2: Initial screen layout

feature vector entries, timestamps of length T are selected: T should be high enough to capture the interesting behaviour for each temporal slice, and as low as possible for a precise detection of the network attack. By manual inspection of the detected anomalies using different timeslice duration T (starting from $T = 60$ to $T = 600$ with a $60s$ step), we found that a 5–minutes ($T = 300s$) slice is a good trade–off. With that slice size, anomalies in the dataset are well temporally localized and the detection rate is maximized. After slicing the data into timestamps, the entropy vectors are computed. According to statistical theory, the feature vectors can be seen as a statistical process that generates a 10–dimensional vectors of random variables for each timestamp, as shown in Table 2. Then, in order to detect the principal orthogonal components of the discrete input signal, the Karhunen–Loeve Transform (KLT) is applied to the realization matrix, and eigenvectors and eigenvalues of the input data are obtained. The complexity of this process using the PCA shown in [37] is $O(d^2)$ (where d is the number of time slices), and the eigenvectors can be incrementally updated [15].

By using a subset of the eigenvectors for the reconstruction step it is possible to create a least square approximation of the original vector (through a dimensional reduction). Let $B = \{\varphi_1, \varphi_1, \ldots, \varphi_{10}\}$ be the complete orthonormal base where the vectors are sorted according with their respectively eigenvalue in descending magnitude order, and $B_k = \{\varphi_1, \varphi_1, \ldots, \varphi_k\}$ be the reduced one. Then a generic observation vector v can be approximated using only k basis as:

$$\tilde{v}_k = \sum_{i=0}^{k} < v, \varphi_i > \cdot \varphi_i = \sum_{i=0}^{k} \left(\sum_{j=1}^{10} v[j] \cdot \varphi_i[j] \right) \cdot \varphi_i$$

where $< v, \varphi_i >$ is the analysis coefficient related to the i–th base.

In order to detect an anomalous behaviour, a metric (measure of distance) is requested. We used the Euclidean norm of the difference vector as an anomaly indicator:

$$A = ||v - \tilde{v}_k||_2 = \sum \left(v(c) - \tilde{v}_k(c) \right)^2$$

By using k components in tlfe analysis base, it is possible to detect anomalies by looking at the A value. If A is small, the reduced base can approximate the input signal well, because the approximated vector is very close to the original one. Conversely, when A is large the approximated vector is substantially different from the original one, which could indicate a network problem (intrusion, virus, etc.).

4. TVI ARCHITECTURE: VISUALIZATION ENGINE

The overall layout of the TVi user interface is shown in Figure 2. On the top (labelled "1" in the image) the timeline of the entropies for every network feature is shown. On the left side there is a checkbox panel, which lets the user select which features to show in the timeline graph on the right. There is also a checkbox for displaying the anomaly value (computed as in Sec. 3.1.3), and a selector for setting the dimensionality of the vectors of the reduced base used in the anomaly detection step. By selecting the anomaly checkbox, an anomaly list is shown in the bottom–left side of the screen. The user can also select all or none of the features. The feature colours were chosen from an existing color map [2] such that they are easy to differentiate when they appear in a nicely ordered sequence (such as a legend). In the timeline, it is possible to select a period of time (win-

dow), and the low–level details of the network status during the selected times are shown in the bottom area (region "2"). It is also possible to shift the observing window of one or more ticks, forward or backward, using the two buttons on the right side of the layout. On the left of the screen a tab selector with the settings of the visualization plane is placed. It contains 3 tabs: a histogram viewer, a graph viewer and a matrix viewer. According to the tab the user selects, more controls become available. The bottom–right side contains the selected visualization.

4.1 Anomalies

When the anomaly checkbox is set, the anomaly list is populated with time periods where the anomaly value is greater than zero, so the user can quickly navigate to periods of suspicious activity. By selecting an item on this list, the anomalous time range is automatically highlighted in the visualization timeline, as shown in Figure 3. If the database contains also the output of another IDS, like Snort or Bro, the user can choose to display it, so as to be able to correlate data from different sources.

4.2 Histograms Visualization

The histogram of a feature presents information about the traffic probability distribution of the selected variable along the selected timestamps. Histograms are actively used in many anomaly detection techniques, as shown in [23] and [24], because they are scalable and effective for visualization. In our system, a histogram is presented for each selected feature. In the histogram, the x–axis contains the unique values of the distribution (discrete events), and along the y–axis the hit rate (i.e. the stochastic probability of that event). The histogram is sorted in reverse height, i.e., the first bin is the one that contains the most hits and the last one contains the least. Empty bins are not shown: for example an unused port is not shown in the x–axis. A checkbox lets the user toggle between a linear scale and a log–log scale (both x–axis and y–axis use a logarithm scale) which is useful for displaying or comparing high–value bins or wide histograms.

To the right of these histograms is another representation of the same information. This representation is different if the selected feature is a port or an IP address. In the former case there is a histogram sorted by port number. Along the x–axis all the ports have a bin, including empty columns. Since there are $64K$ ports, the x–axis is scaled logarithmically, because low–value ports have greater relevance than high–value ones, especially in the range $(1, 1024)$. In the latter case, when an IP feature is selected, a Hilbert map is shown. The Hilbert transform maps an IP address to a point in a 2–dimensional space. The benefits of this approach are discussed in [36], but the key feature is about localization: similar IP addresses (e.g., addresses of the same subnet) are mapped nearby in the 2D space.

When more than one timestamp is selected in the timeline, the normal histogram views are not shown. Instead, the histograms for each timestamp are visualized in a 3D space, where the third axis (z–axis) represents the time. However, this 3D space is projected orthogonally, such that the x and y axes are kept horizontal and vertical, respectively, and no distortion is incurred due to depth effects. All these graphs are interactive: by clicking on one bin or on one line of an histogram, more specific information is displayed.

4.3 Graphs Visualization

In order to investigate details of anomalous activity, we employ a graph representation. Graphs offer a very intuitive visualization mean. The user can directly see the connections between each node in the network, and thus easily identify anomalous actors and their behaviour. The goal of the graph visualization is to show the interactions among hosts. Each node in the graph corresponds to a distinct IP in the network, and an edge represents the connection between the two hosts. Colour is used to show directionality: the red side of the link is the source; the green side is instead the destination. In the literature "Traffic Dispersion/Activity Graphs" (TDG or TAG) (as shown in [20], [22] and [21]) are actively used for visually monitoring application flows, for classifying applications and protocols and for detecting attacks, intrusions or anomalous behaviours.

We use the "Fast Multipole Multilevel Method" (or FM^3) for graph layout [19]. It is a relatively fast algorithm, with a worst case running time of $O(|V| \log |V| + |E|)$ (where $|V|$ and $|E|$ are respectively the vertex set and the edge set cardinality) and with a GPU implementation this method can run 20 time faster than a normal CPU implementation [17]. Furthermore, it clearly visualizes the structures of the data.

Combined with this classical graph visualization, we also use a geolocalized map. It arranges IPs based on their latitude and longitude. TVi uses for this purpose the hostip [5] database, which we replicated on our local SQL tables. We use the Mollweide projection to place the IPs on the world map.

This visualization window is also highly interactive: selecting a point or a link makes TVi display information about it, exclude the background, enlarge the most populated point (points with more than 1 IP address), and filter the edges. In such ways the interactive map can be very useful for analysing anomalies.

4.4 Matrix Visualization

If the graph becomes too dense to be interpreted, TVi also features matrix visualization. With a matrix it is possible to correlate two features of the network even when it is densely populated. In TVi the user can select a protocol, filter the ports (if applicable, i.e., with TCP and UDP) and decide which network features will be represented on the two axes. The axes can be set to source IP address, destination IP address, source port, or destination port (if applicable). The value contained in each cell of the matrix can be a binary value (1 for a connection between row and column, 0 otherwise), or the number of bytes among the row value and the column value, or the connection flows cardinality. In order to visualize the value a colormap is used, which the user can select from some included colormaps or customize as desired. To reorder the matrix, for better visualization of the patterns, we use the Sugiyama [28] algorithm. By reordering the rows/columns of the matrix, patterns contained inside the matrix can be more readily identified (IP sweep, portscan, DoS, etc.).

4.5 External Detail Tools

When TVi is fed `pcap` traces as input, it is also capable of interacting with external packet–level visualization tools. Through its the visual interface, the user can export and load the selected trace in Wireshark [11] or Etherape [4]. With these programs active on selected timestamps of the

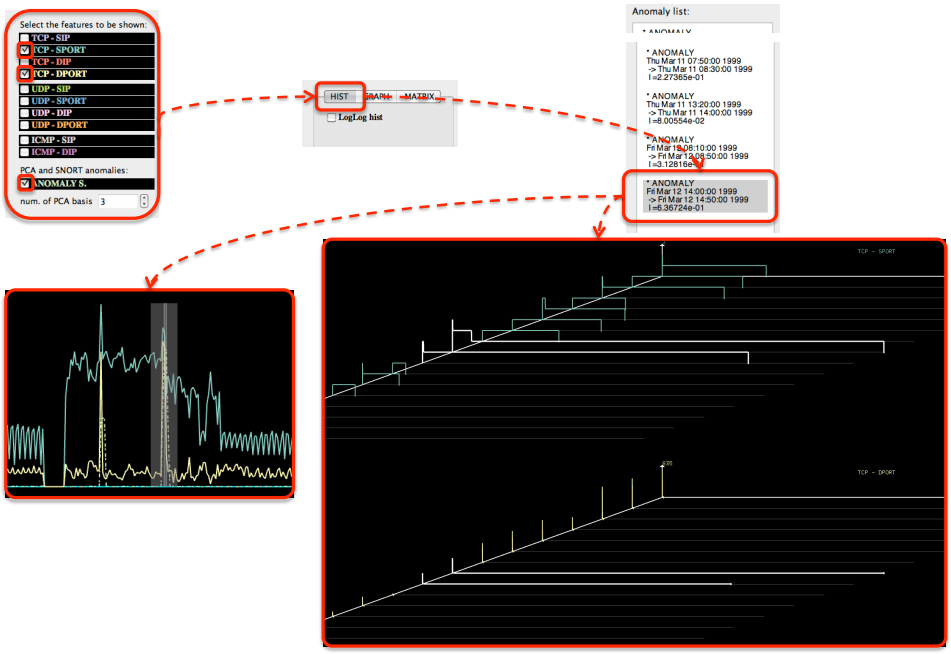


Figure 3: Steps needed to select an anomaly from the list and its histogram visualization

trace the user can investigate specific timing information as desired. This is the zero–level layer (i.e., without abstraction) of TVi.

5. CASE STUDIES

In order to demonstrate the effectiveness of TVi, we have applied it to two datasets: one is old but includes a ground truth of the attacks contained (DARPA), while the other is new and does not contain any ground truth, as it is based on real Internet traffic.

5.1 Datasets

The first dataset used in the case study is the (in!)famous DARPA dataset, provided by DARPA-MIT in 1999 [3]. It is a synthetic database, because it was recorded in a simulated network with traffic generators and no connection to the Internet. Because of this and other shortcomings, it has collected much criticism [30]. Despite this, to this day it remains the only publicly available dataset with ground truth. For this case study we used the second week of data, which contains labelled attacks.

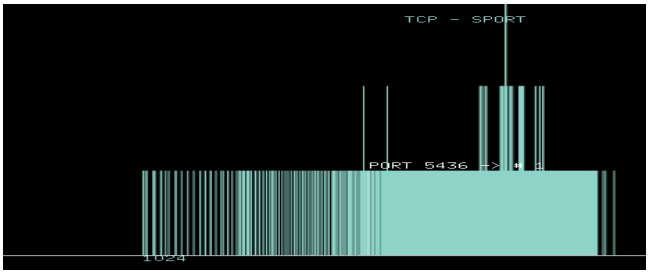
The second dataset was taken from the edge router of the Faculty of Engineering of the University of Brescia. It is a 30 hour trace that contains only the headers of the traffic packets - about 60 GB in total. It was captured between March 17 2009 13:56:35 (Tuesday) to March 18 2009 16:47:35 (Wednesday). The edge router forms the border between the internal network and the rest of the Internet, so all incoming and outgoing traffic passes through it. There are several thousand users on the campus network every given work day.

5.2 Case Study - DARPA 1999 Dataset

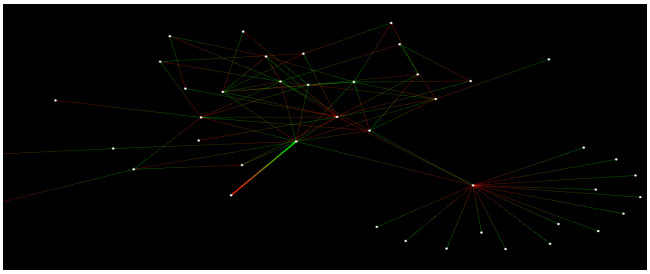
Using TVi to detect and analyse anomalous activity requires following a few steps, as depicted in Figure 3. Initially the screen is empty, i.e., no features are selected. The user can now add or remove the feature timelines, by selecting or un-selecting the checkboxes. Here, we select the TCP source and destination ports as well as the anomaly metric. Then, by selecting an anomaly from the list tab, the related timestamps are highlighted in the timeline, and multiple histograms are shown. These histograms represent the statistical distribution of the feature during the selected timesteps. In this example, we can see that the histograms in the central point of the highlighted windows have a larger support. This means that more distinct values of the features have been used in the temporal window. We can thus hypothesize that it is a portscan attack, where one or more host tried to connect to many destination ports of the target host over TCP.

By looking at central histogram, we can then see that many source and destination ports have been used (see Fig. 4(a)). After this, we consider the graph of the connections (Figure 4(b)) where we note that the thickest connection in the graph is between two hosts, identified by the IP addresses 172.16.112.50 and 209.167.99.71 (the former is the destination, the latter the source). At this point we can argue that this could be a portscan, where 209.167.99.71 is the attacker, and 172.16.112.50 is the destination host. Figure 4(e) also shows that the attacker is located in Ontario, Canada and the target is an inside host, as its IP belongs to the private address space of the home network.

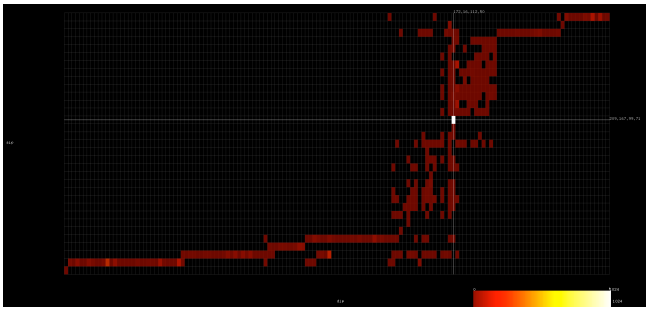
Another representation of this event can be seen using the matrix representation of the attack. From Figures 4(c) and 4(d), we can easily recognize that a portscan event is in progress. From Figure 4(c) we can detect that the highest value of the matrix is the connection between the IP 172.16.112.50 and 209.167.99.71, and from the Figure 4(d) we can see that a portscan is done on the second target. At this point we can be sure that we have detected a portscan



(a) Histogram distribution of the source ports, TCP, during the detected anomaly. As it can be seen, high ports have been contacted sequentially one or more times (the peak value is 3). The histogram of the destination ports is similar.

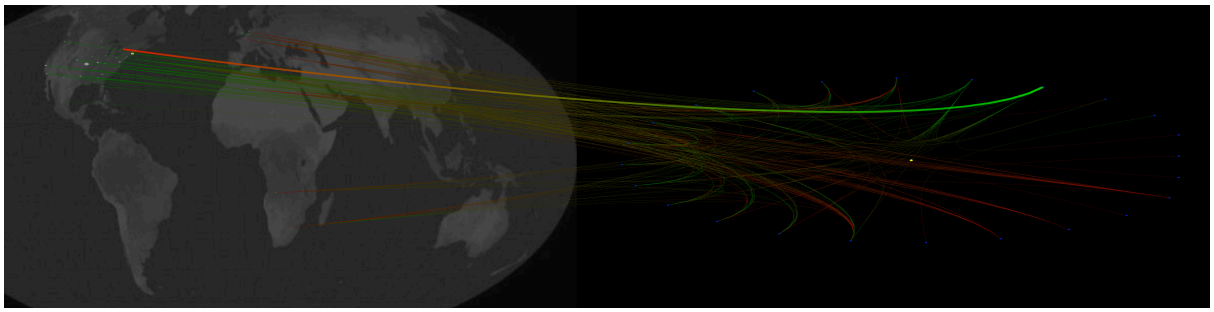


(b) Graph during the observed attack. White points are the hosts, links are the connections among hosts (red is source; green the destination).



(c) Matrix representation during the observed attack: on the x axis the destination IPs are shown and on the y axis the source IPs

(d) Matrix representation during the attack: in this case on the x axis there are the ports. We can easily see the scan made by one host.



(e) World–graph during the observed attack. Each IP is geolocalized using the GeoIP database. The yellow point is the home network, and the blue ones the internal hosts

Figure 4: Steps for anomaly detection in DARPA 1999 dataset

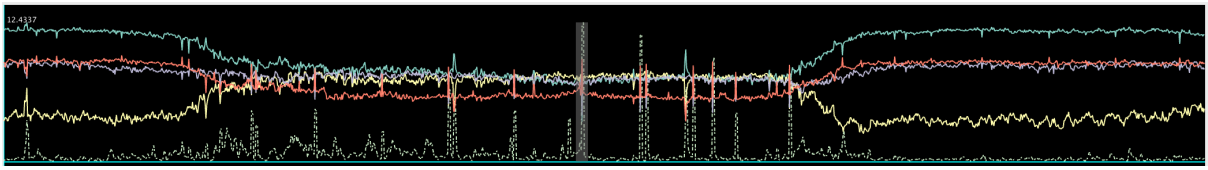
attack, that 172.16.112.50 is the destination IP, and the initial timestamp is *Friday 03/12/1999 14.10.00*. According to the ground truth information in DARPA's database, we can easily confirm our hypothesis: *On 03/12/1999, at 14.13.10 the IP* 172.16.112.50 *was the destination of a portscan attack.*

5.3 Case Study - UniBS 2009 Dataset

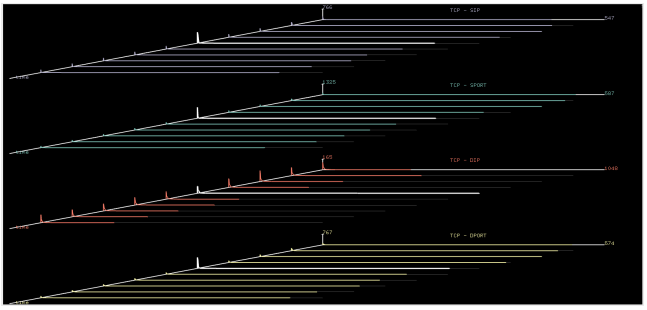
In the anomaly list an entry is selected, as highlighted in Figure 5(a). We note that it is correlated with activity in the TCP-related features, so we activate all four of them. During this time window we note some facts in the histogram visualization (see Figures 5(b) and 5(c)):

1. The source IP histogram has the first two bins higher than normal (the previous histograms). The IP addresses are 222.45.X.Y and 122.227.Z.K (last two bytes anonymized for privacy reasons).
2. The source port histogram has a high first bin. Its port is 6000.
3. The destination IP histogram is wider than normal.
4. The first and the second bins of the destination port histogram are higher than the normality (ports 2967 and 1433).

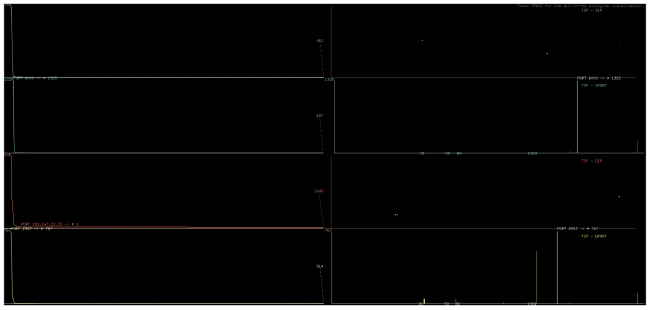
From this we can guess that the attack is an intruder from two IP addresses who tries to scan multiple IP addresses using the same source and destination ports. With the help of the graph representation we can confirm our idea. By using a filter on the port 6000 and then on the port 1433 we disco ver that both attackers use the source port 6000 (ref. Figure 5(d) and 5(e)). In these views we can see that the IP address 222.45.X.Y scans the port 2967 and the attacker 122.227.Z.K the port 1433. So the attack is from two scanners that are trying to connect to the same port on



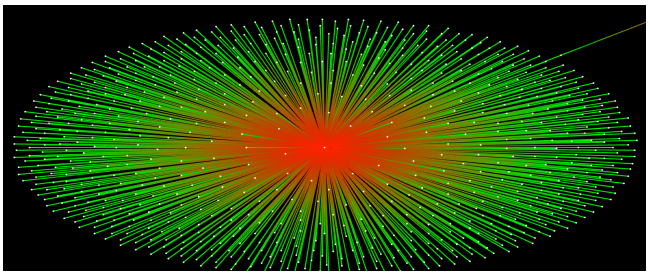
(a) Timeline of the UniBS09 dataset showing TCP features, the anomaly values and the selected window



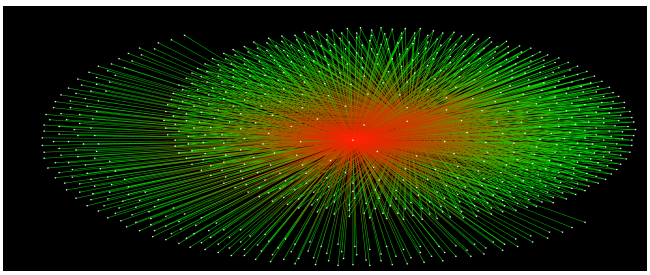
(b) Histogram distribution during the anomaly peak



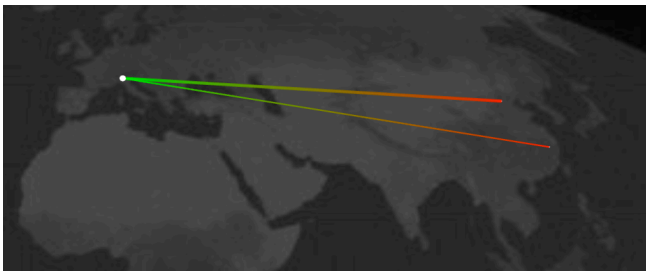
(c) Histogram distribution in the middle of the anomaly



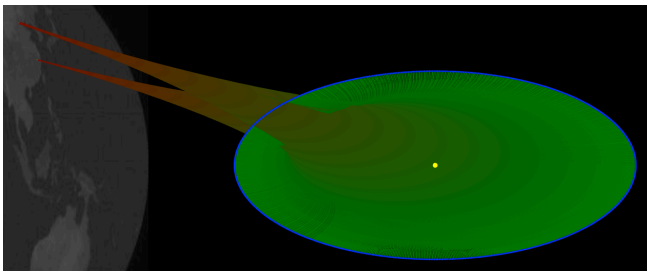
(d) Graph of the network filtered on port 2967. The point in the centre represent the IP 222.45.X.Y



(e) Graph of the network filtered on port 6000. The central points are the IPs 222.45.X.Y and 122.227.Z.K



(f) World graph of the network filtered on port 6000. The left point represents the attacked point (multiple public IP addresses of the University of Brescia), and the two points on the right the two attacker, from China



(g) World graph of the network filtered on port 6000. We can easily see that the two hosts are trying to contact multiple IPs of the same geo–located network. In this image the yellow dot is the University of Brescia's network, and the blue dots are the hosts inside its campus

Figure 5: Steps for anomaly detection in UniBS 2009 dataset

multiple target machines. Using the world map layout we can see that the two attackers (see 5(f), and 5(g)) originate in China and are trying to scan public IPs in the University of Brescia's network.

Finally, we search for this attack in Snort's output and notice that only two entries in the Snort alert file are related to the attack: one alert is a TCP portscan to one IP (with priority 3, the least value) and one alert is a "TCP/IP message flooding" directed to another IP. Neither of these alerts accurately describes the pattern of the entire attack and both would likely be ignored by analysis of the Snort log. Hence, by relying on the Snort log we would have the wrong idea about the IPsweep attack that occurred on the network, but our system accurately detected and visualized the attack. After further analysis on specialized IDS forums, we determined that the detected attack is called a *DoS HGOD SynKiller Flood*.

6. EXPERIMENTAL RESULTS AND AVAILABILITY OF TVI

TVi is written in C++ using OpenGL, QT, OGDF [7] and

MySQL. The time needed for detecting and understanding typical anomalies, as the two explained in the previous section, in the network is less than one minute for trained users. New users might take some more time to get acquainted to TVi.

Our results show that both on a synthetic trace with ground truth and a recent trace coming from a production network TVi is capable of significantly reducing the time and effort required to detect anomalies, especially the ones such as scans that are the first step towards an actual attack.

Unfortunately, neither the DARPA trace nor the UniBS one contains actual, dangerous attacks. Therefore, even though we are confident that the TVi architecture would be able to show its advantages over existing mechanisms even in those cases, the unavailability of public traces with actual attacks makes it virtually impossible to test these cases fully.

In order to solve this issue, and to contribute to the community, we released TVi [10] with an open source license (GPL). We want to encourage practitioners and academics to put it at use, and report back to a (public) database on TVi's webpage about its capabilities of making the detection of actual attacks easier.

7. CONCLUSIONS

This paper introduces TVi, a visual, interactive tool for network monitoring and anomaly detection. The efficient and novel anomaly detection metric based on the K-L transform allows the system to rapidly direct users to time periods of anomalous activities. The integrated visual representations provide an intuitive mean for detailed analysis of such activities. At that point, the integration with Snort and other monitoring utilities allows users to pinpoint with great precision the sources and behavior of anomalous activities.

Furthermore, being based on modern DBMS infrastructures, TVi has the potential to scale to very large network monitoring tasks. The combination of these features allow for a much shorter response time to attacks than either a traditional NIDS such as Snort or standalone forensic analysis visualizations.

The effectiveness of TVi has been preliminary proven on two heterogeneous datasets, although certainly further work is in order to fully assess its capabilities. One important step in this direction is the release of TVi [10] under an open source license, which allows the community to test it on their network, without resorting to experimenting with public anonymized traces.

8. REFERENCES

[1] Cisco netflow protocol. http://www.cisco.com/web/go/netflow.

[2] Color brewer 2.0, color advice for cartography. http://colorbrewer2.org/.

[3] Darpa 1999 dataset. http://www.ll.mit.edu/mission/communications/... .../ist/corpora/ideval/data/index.html.

[4] Etherape, a graphical network monitor. http://etherape.sourceforge.net.

[5] Hostip info ip addresses database. http://www.hostip.info/dl/index.html.

[6] Mysql cluster. http://www.mysql.com/products/cluster/.

[7] Ogdf, c++ open graph drawing framework. http://www.ogdf.net.

[8] Snort, a free lightweight network intrusion detection system. http://www.snort.com.

[9] Tnv: The network visualizer. http://tnv.sourceforge.net/.

[10] TVi: the UniBS Trace Visualizer. http://www.ing.unibs.it/ntw/tools/tvi.

[11] Wireshark packet analyzer. http://www.wireshark.org.

[12] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou. Network anomaly detection and classification via opportunistic sampling. *IEEE Network*, 23(1):6–12, January/February 2009.

[13] A. Aris, B. Shneiderman, C. Plaisant, G. Shmueli, and W. Jank. Representing unevenly-spaced time series data for visualization and interactive exploration. In M. Costabile and F. PaternÃš, editors, *Human-Computer Interaction - INTERACT 2005*, volume 3585 of *Lecture Notes in Computer Science*, pages 835–846. Springer Berlin / Heidelberg, 2005.

[14] G. Conti, K. Abdullah, J. Grizzard, J. Stasko, J. A. Copeland, M. Ahamad, H. Owen, and C. Lee. Countering security analyst and network administrator overload through alert and packet visualization. *IEEE Computer Graphics and Applications*, 26(2):60–70, 2006.

[15] I. Dagher. Incremental pca-lda algorithm. *Computer Science Journals*, 2010.

[16] J. Glanfield, S. Brooks, T. Taylor, D. Paterson, C. Smith, C. Gates, and J. McHugh. Over flow: An overview visualization for network analysis. In *International Workshop on Visualization for Cyber Security, VizSec 2009*, pages 11 –19, October 2009.

[17] A. Godiyal, J. Hoberock, M. Garland, and J. C. Hart. Rapid multipole graph drawing on the gpu. In *Graph Drawing'08*, pages 90–101. 2008.

[18] J. Goodall and M. Sowul. Viassist: Visual analytics for cyber defense. In *Proceedings of the 2009 IEEE Conference on Technologies for Homeland Security*, HST'09, pages 143–150. IEEE, 2009.

[19] S. Hachul and M. Junger. An experimental comparison of fast algorithms for drawing general large graphs. In P. Healy and N. Nikolov, editors, *Graph Drawing*, Lecture Notes in Computer Science, pages 235–250. Springer, 2006.

[20] M. Iliofotou, P. Pappu, M. Faloutsos, M. Mitzenmacher, S. Singh, and G. Varghese. Network monitoring using traffic dispersion graphs (tdgs). In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 315–320, New York, NY, USA, 2007. ACM.

[21] Y. Jia, J. Hoberock, M. Garland, and J. Hart. On the visualization of social and other scale-free networks. *IEEE Transactions on Visualization and Computer Graphics*, 14:1285–1292, 2008.

[22] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *SIGMETRICS '09: Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, pages 49–60, New York, NY, USA, 2009. ACM.

[23] A. Kind, M. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110 –121, 06 2009.

[24] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 217–228, New York, NY, USA, 2005. ACM.

[25] K. Lakkaraju, W. Yurcik, and A. J. Lee. Nvisionip: netflow visualizations of system state for security situational awareness. In *ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, pages 65–72. ACM, 2004.

[26] S. Lau. The spinning cube of potential doom. *Communications of the ACM*, 47:25–26, June 2004.

[27] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das. The 1999 darpa off-line intrusion detection evaluation. *Elsevier Computer Networks*, 34(4):579–595, 2000.

[28] E. Makinen and H. Siirtola. Reordering the reorderable matrix as an algorithmic problem. In *Theory and Application of Diagrams, Lecture Notes in Computer Science*, volume 1889, pages 453–468. Springer, 2000.

[29] R. Marty. *Applied Security Visualization.* Addison-Wesley Professional, 2008.

[30] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security*, 3:262–294, November 2000.

[31] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: a tool for port-based detection of security events. In *ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC '04, pages 73–81, 2004.

[32] M. Navas and C. Ordonez. Efficient computation of pca with svd in sql. In *Workshop on Data Mining using Matrices and Tensors*, pages 1–10, New York, NY, USA, 2009. ACM.

[33] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *ACM SIGCOMM conference on Internet measurement (IMC 08)*, pages 151–156, 2008.

[34] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Elsevier Computer Networks*, 51:3448–3470, August 2007.

[35] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. In *ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '07, pages 109–120, New York, NY, USA, 2007. ACM.

[36] T. Samak, S. Ghanem, and M. Ismail. On the efficiency of using space-filling curves in network traffic representation. In *IEEE INFOCOM Workshops 2008*, pages 1–6, April 2008.

[37] A. Sharma and K. K. Paliwal. Fast principal component analysis using fixed-point algorithm. *Pattern Recogn. Letters*, 28(10):1151–1155, 2007.

[38] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks, and J. McHugh. Flovis: Flow visualization system. *Conference For Homeland Security, Cybersecurity Applications and Technology*, pages 186–198, 2009.

[39] A. Yelizarov and D. Gamayunov. Visualization of complex attacks and state of attacked network. In *International Workshop on Visualization for Cyber Security*, pages 1 –9, october 2009.