Supporting Intrusion Detection by Graph Clustering and Graph Drawing

Jens Tölle[1], Oliver Niggemann[2]

[1] University of Bonn,
Römerstraße 164, D-53117 Bonn, Germany
toelle@cs.uni-bonn.de

[2] University of Paderborn,
Warburger Straße 100, D-33100 Paderborn, Germany
murray@uni-paderborn.de

Abstract: This paper presents a description of a system supporting the detection of intrusions and network anomalies by analyzing and visualising traffic flows in computer networks. The system supervises the typical communication structure in the network and acts as an anomaly detection component of an Intrusion Detection System. Events are generated in the case of sudden variations of the traffic structure. Visualization of the traffic structure is used to help the security manager to gain an overview on the current traffic structure and to help identifying the type and the location of network anomalies.

Keywords: anomaly detection, graph clustering, graph visualization

1. Introduction

Network and security management has to assure uninterrupted access to the communication infrastructure. With growing networks and increasing amount of transported data, it gets more and more complicated to supervise the operation of the communication systems.

Sometimes computer networks are not well protected against attacks from the outside, so additional surveillance may be necessary. But even well protected networks need surveillance. A lot of these networks are threatened from the inside. Intrusion Detection Systems (IDSs) help securing these networks. This paper focuses on a tool for visualizing and detecting anomalies of the traffic structure.

Several distributed Denial of Service attacks (for example [1]) have shown the necessity of better protecting computers and networks connected to the Internet. Due to widely available attack tools [2], attacks of this kind can be carried out by persons without in-depth knowledge of the attacked system. Insufficient protected open university networks are an example for networks that need additional surveillance.

These networks often include vulnerable computers and offer high bandwidth connections to the Internet. These features are the reason why attackers are interested in these networks. The machines in these networks are not the goal of the attacks. Normally, they do not contain interesting information for the attacker, but they are suitable for scanning other networks and starting (for example) Denial of Service attacks.

The system described below has two different purposes. On the one hand, it is possible to analyze typical traffic patterns and to create a general view on the network usage. The visualization of these data presents the network manager the information needed for decisions on future modifications of the network topology and exchange of network equipment. On the other hand, the collected information on traffic patterns presents the security manager an up-to-date view on the current network usage. It is possible to detect anomalies automatically and to create events for Intrusion Detection Systems. This paper will focus on the second purpose. We believe, that graph clustering delivers patterns that make it possible to visualize and automatically detect anomalies in the network traffic.

We have chosen a graph based approach to detect anomalies giving evidence of an attacker in the network. The idea of using graphs as an abstraction for network traffic has been successfully tested in different Intrusion Detection Systems. Graph based methods are for example used in GrIDS [3], a system developed at the Department of Computer Science of the University of California at Davis.

2. System Description

In this section we describe the main parts of the system. It consists of a packet collecting unit, a graph construction and clustering unit, the visualization module and the event generating module.

2.1 Data collection

The data needed for the system is collected in the supervised network. This work can either be done by one of the existing computers or by dedicated hardware, for example RMON/RMON II devices. Data collection in a network equipped with Ethernet hubs is easy, while the collection in an switched network requires a configuration of the switch that copies each packet to the port of the collecting device. The system was both tested in an environment equipped with Fast Ethernet hubs and in a switched Fast Ethernet network.

Data source in both environments was an IP sniffer, running on an standard PC. This machine has to be fast enough to catch up with the speed of the network delivering the packets. Another requirement was enough hard disc memory to store detailed information about the collected packets.

After collecting all the packets during a time interval, filtering for different protocols can be done. The resulting data is delivered to the graph construction unit.

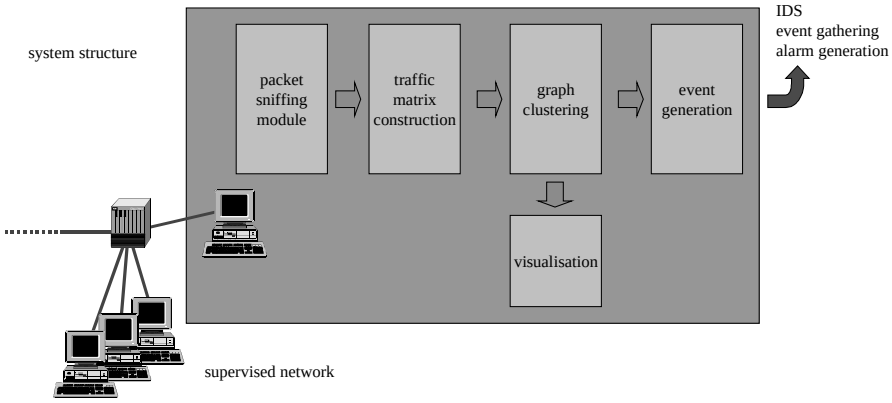

Figure 1: system structure

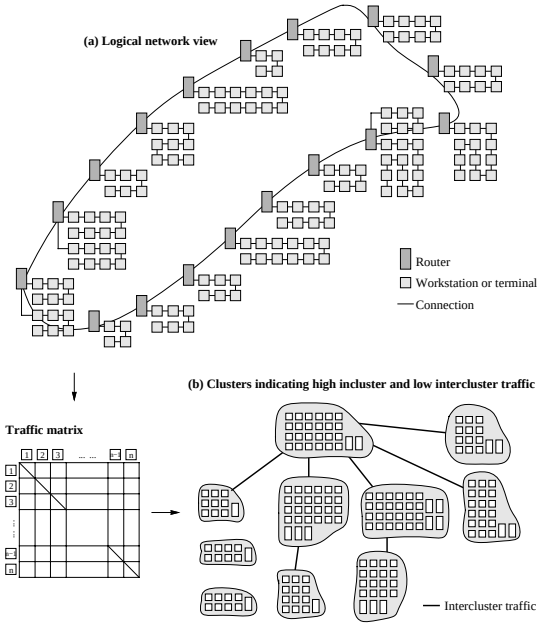

Figure 2: Communication clusters identified in the traffic matrix of a computer network; order of magnitudes, the network structure, as well as the identified cluster-graph rely on real-world data.

2.2 Graph Construction and Clustering

A graph is used as an abstraction of network traffic. Computers are represented by nodes, communication between computers is indicated by edges, weighted by the amount of exchanged data. Figure 2 shows such a clustering of a network. In our system, graph clustering algorithms are applied to the traffic graph. Clustering

decomposes a graph G=(V,E) into clusters $C_i \subseteq V$, $0 \leq i \leq n-1$, where $C_1 \cup C_2 \cup ... \cup C_n = V$ and $\forall\ 0 \leq i,j \leq n-1$: $C_i \cap C_j = \emptyset$. This decomposition should represent the internal structure of the graph. In the domain of network traffic, a cluster should consist of nodes with a high inter-node traffic.

Clustering data given as graphs has been a focus of research for years. In order to give an overview, existing algorithms are classified according to several criterions as follows:

- hierarchical versus non-hierarchical algorithms
  Hierarchical algorithms create a hierarchy of clusters by continuous subdivision or union of clusters. In order to obtain a unique clustering, a second step transforming the hierarchy into a clustering becomes necessary.
  These algorithms can be further classified into divisive and agglomerative approaches:

  Divisive algorithms start with each vertice being its own cluster and union clusters iteratively. For agglomerative algorithms on the other hand the entire graph forms initially one single cluster which is successively subdivided. Examples for divisive algorithms are min-cut-clustering ([4],[5]) or dissimilarity-based algorithms (e.g. [6]). Typical agglomerative algorithms are k-nearest-neighbours or linkage-methods ([7],[8],[9]).

  Non-Hierarchical algorithms subdivide the graph into clusters within one step. Examples are clustering techniques based on Minimal-Spanning-Trees ([10]) or self-organizing Kohonen networks (e.g. [11]) .

- exclusive versus non-exclusive algorithms
  Exclusive clustering-algorithms assign every node to precisely one cluster, while non-exclusive algorithms assign a node to several clusters. All algorithms mentioned above are exclusive, an examples for a non-exclusive algorithm is fuzzy-clustering ([12]).

- algorithmic versus descriptive methods
  While algorithmic methods (e.g. all algorithms mentioned above) compute the clustering by means of an algorithm, do descriptive methods define a criterion for a clustering's quality. The result is found by optimizing this criterion. This approach is used in [13],[14],[15].

Most of the algorithms mentioned above rely on extra parameters in order to compute the number of clusters. Since neither the number of clusters nor any other features are known about the clusters in network traffic, only few cluster algorithms can be applied:
Methods optimizing a criterion are able to determine the number of clusters, but the optimization process forms a time-consuming step, making it unsuitable for real-time network analysis. For all these reasons we decided to use the relatively new MajorClust-Method (for details see [16]). MajorClust will now be described briefly:

MajorClust is exclusive and algorithmic. Initially, the algorithm assigns each node of a graph its own cluster. Within the following, agglomerative re-clustering steps, a node adopts the same cluster as the majority of its neighbors belong to. If there exist several such clusters, one of them is chosen randomly. If re-clustering comes to an end, the algorithm terminates.

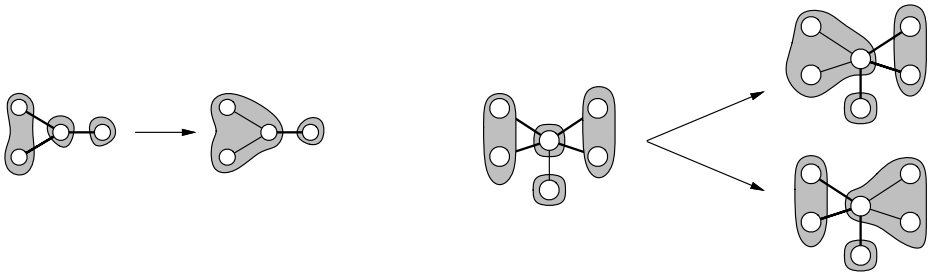

Figure 3: A definite majority clustering situation (left) and an undecided majority clustering situation (right)

The left-hand side of Figure 3 shows the definite case; most of the neighbors of the central node belong to the left cluster, and the central node becomes a member of that cluster. In the situation depicted on the right-hand side, the central node has the choice between the left and the right cluster.

The result is a simplified graph, containing information on the typical traffic structure of the network. The typical result are clusters representing working groups, working together in projects and therefore exchanging larger amounts of data and clusters, representing client and server relations. These relations lead to star like topologies, with the video-, file-, or web server being the root of the star.

2.3 Graph Visualization

There are several possibilities to present the results of the graph clustering. An easy to implement way is the presentation of the results of the clustering process in lists. This form of presentation has several disadvantages. It is not possible to achieve a general overview on the current traffic structure in the network. Modifications of the structure are hard to discover in the lists.

Due to this, a graphical representation of the clustering results was chosen. Visualization of network traffic is an important task for planning an managing large networks. Research is done on this item to provide network managers with an insight to the usage of their systems. Most of these systems, for example [17],[18] present traffic information in a geographical way. The visualization of the traffic depends on the location of the nodes. In our system, we've chosen to present the results of the traffic analysis in another way. Our focus is on the structure of the traffic, and therefore we group nodes with strong communication relations together without considering their geographic location. This leads to an easy to understand presentation

of the current traffic. This graphical representation has the additional advantage to clearly show modifications of the network behavior.

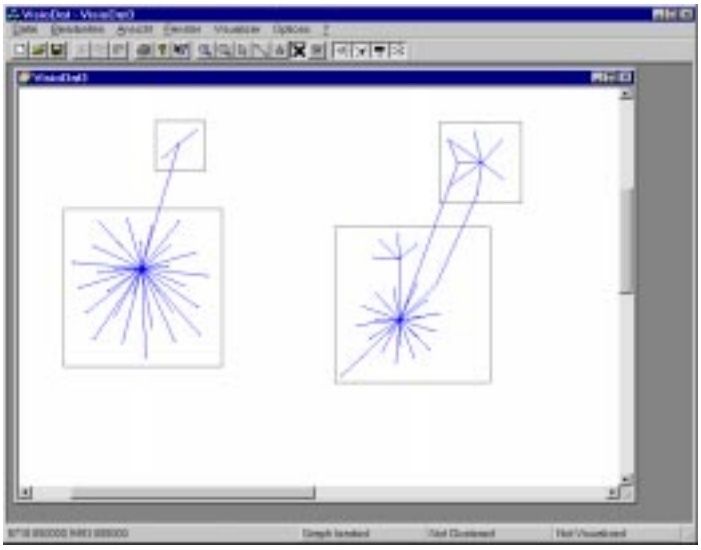

Figure 4: A screenshot with a visualization of a traffic structure in a network

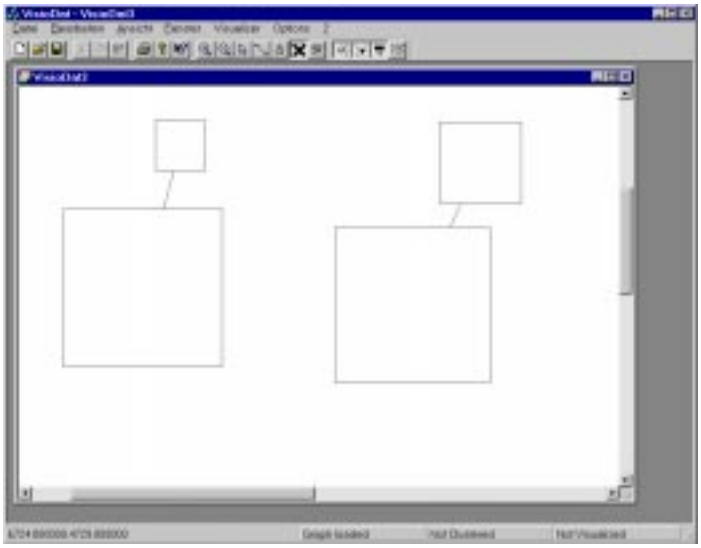

Figure 5: The same traffic structure, only inter-cluster-edges drawn

A visualization method has been developed that emphasizes the graph structure as found by the clustering step. This method mainly uses well-known graph-drawing algorithms. Graph-Drawing is the task of drawing a given graph on a plane. For

visualizing network-traffic the well-known Spring-Embedder (see also [19], [20], [21]) methods has been implemented. This method is specially suited for typical network graphs. Other methods, e.g. hierarchical layout (see [22]), have also been implemented and evaluated, but for network traffic graphs they proved to be inappropriate.
Our tool first places the clusters on the plane (figure 3). The reader may note that the clusters form a new graph. Two clusters are connected if nodes within those clusters are connected by an edge. The clusters can be positioned using the Spring-Embedder. In a second step the nodes with the clusters are placed. This can be seen in figure 4 and 5.
This visualization helps the security manager to build his own opinion on messages from the event generator described in the next section. A major problem in intrusion detection systems are so called false negatives and false positives. The problem of any anomaly detection system is the fact, that anomalies in the behavior are not necessarily based on intrusions (causing false positives) and intrusions do not necessarily cause anomalies in the system (leading to false negatives). Our visualization helps discovering false positives (alarms without attack).
It is easy to see, that there are more reasons for modifications in the traffic structure. Changes in network topology, new network devices, start or end of projects are examples for reasons for modifications in the typical structure. The user of the visualization module is able to use his additional personal knowledge to decide on the results of the event generating process.
A special benefit of our visualization is the possibility of displaying modifications of the traffic structure in consecutive traffic matrices. Position and color of nodes indicate changes in their membership of different clusters. It is easy to track varying cluster, giving information on nodes that have changed their communication behavior. It is even possible to present longer series of traffic matrix visualizations in the form of an animation, showing long term behavior of the communication structure.

2.4 Event Generation

Our system collects live network traffic and applies a clustering algorithm to the traffic graph. Additional information on the structure of the resulting clusters are calculated for the event generating process. Features $v(G)=\{v_1,...,v_p\}$ are used for the description of the state of the current network $G=(V,E)$:

- number of communicating network nodes and number of found clusters
- minimum, maximum and average degree of the nodes in the communication graph
- minimum, maximum and average number of nodes in a single cluster.
- minimum, maximum, average and total number of internal edges in each cluster of the communication graph. Internal edges represent internal communication in a cluster. Source and destination of a communication relation are nodes of the same cluster.
- minimum, maximum, average and total number of external edges of the cluster. External edges are edges with source and destination node not belonging to the same cluster.

Modifications in the traffic graph are new nodes, lost nodes, splits of clusters, merging of clusters, and nodes moving from one cluster to another.

A function q: R* -> N mapping from feature vectors onto intrusion methods can now be learned by means of regression. For this a number of typical intrusion situation and some non-intrusion situations have to be given.
For each situation i the traffic Graph G_i and the feature vector $v(G_i)$ is created. Since q(v(Gi)) is known (either 0 for "no intrusion" or an identity number for the intrusion), a supervised-learning strategy (e.g. neural-nets, see also [23]) or regression (see [24], [25]) can be applied to learn this function.

Experiences with the event generation
This section presents practical experience with the system. Experiments were made in two different environments. The first environment was a test cluster of workstations which was exclusively used for the experiments. First tests were done in this network, to check whether the system is able to detect artificial communication between the workstation in the network. These tests were done to have the possibility to reproduce the conditions of our experiments exactly.
The other environment was the switched network of a student laboratory in regular use. One of the switch ports was configured to copy all the traffic to a workstation running the sniffing module. This environment was used to gather real data in order to have the possibility to learn about the system in a real network.
Log files were collected during normal network usage. To learn about anomalies occurring during attacks, different attack tools were tested while the system was running. The visualization of the traffic during some of the attacks showed anomalies, depending on the kind of the attack. It is clear to see, that this method is only able to detect intrusions producing a considerable amount of network traffic.
Here are some of the results from our tests with live data from the students laboratory:
In order to generate an event in the case of these anomalies, learning techniques were used to help looking for typical patterns. We used regression to learn the function indicating whether the current network state is caused by an intruder or not.

Regressions of the function deciding whether an intrusion occurred were made with 114 test data sets. 57 of these data sets contained intrusion tests, 57 of them were normal network traffic.
One result of a regression is the importance of the different features to detect intrusions. The most important two values found here are the 'average node degree' and the 'average number of external edges'. That means the learning method relies mainly on average values of cluster properties when deciding whether an intrusion happens. This behavior is not always sufficient for distinguishing attacks from normal system usage in the data used for learning, and shows poor performance in analyzing new data.

Experiments with different data sets had a reasonable error rate in distinguishing normal traffic from traffic influenced by intrusions, when testing with the data sets used for learning. Tests with new data sets did not yet lead to satisfying results, leading to both false positives and false negatives. Automatic pattern recognition is

difficult in this domain. The experiments described here are only a first step, that will be followed by further research.

3. Conclusions

The visualization helps the security manager to get insight to the current usage of the computer network. He has the possibility to learn more about the reasons for events or warnings from his intrusion detection system. This form of presentation helps detecting false positives. An additional benefit of the visualization of the traffic patterns is the help in reviewing the existing network topology and planning modifications or enhancements of the current topology.

The event generating system has to be improved. The concepts are interesting, but additional work is needed to optimize the process. The system is still early work. It is possible to automatically detect anomalies in the communication structure of a surveyed network, but the goal of detecting a large number of different attacks is not yet reached. The fact that some of the attacks could be discovered by the system without any knowledge on the used attack techniques encourages us to further research. It is easy to see that this approach to intrusion detection only is appropriate for intruders causing an significant traffic in the supervised network.

4. Further work

An integration in the existing intrusion detection system is planned for the near future. Additional graph algorithms, especially clustering algorithms will be tested and compared with the used ones. More features shall be extracted from the clustered traffic graphs and different learning methods will be tested.
The visualization will be optimized to the needs of the permanent usage of the system. The integration in a larger ID system requires an appropriate correlation of the events of the different event generating systems in order to optimize co-operation of these systems.

References

[1] D. Dittrich, Presentation on Distributed Denial of Service attacks, CERT Distributed-Systems Intruder Tools Workshop, November 1999

[2] D. Dittrich, Web-Page with papers on different attack tools, http://www.washington.edu/People/dad

[3] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, J. Rowe S. Staniford-Chen, R. Yip, D. Zerkle. The design of GrIDS: A Grpah-based Intrusion Detection System, http://seclab.cs.ucdavis.edu/arpa/grids, 1999

[4] T. Lengauer, Combinatorical algorithms for integrated circuit layout, Applicable Theory in Computer Science. Teubner-Wiley, 1990.

[5] Z. Wu and R. Leahy, An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, November 1993.

[6] P. MacNaughton-Smith, W.T. Williams, M.B. Dale and L.G. Mockett. Dissimilarity analysis. Nature 202, 1964.

[7] K. Florek, J. Lukaszewiez, J. Perkal, H. Steinhaus and S. Zubrzchi. Sur la liason et la division des points d'un ensemble fini. Colloqium Mathematicum, 1951.

[8] P. H. A. Sneath. The application of computers to taxonomy. J. Gen. Microbiol. 17, 1957.

[9] S. C. Johnson. Hierarchical clustering schemes. Psychometrika 32, 1967.

[10] C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. IEEE Transactions on computers Vol. C-20, No. 1, 1971.

[11] T. Kohonen. Self Organizing and Associate Memory. Springer-Verlag, 1990.

[12] J.-T. Yan and P.-Y. Hsiao. A fuzzy clustering algorithm for graph bisection. Information Processing Letters 52, 1994.

[13] T. Bailey and J. Cowles. Cluster definition by the optimization of simple measures. IEEE Transactions on Pattern Analysis and Machine Intelligence, September 1983.

[14] R. Sablowski and A. Frick. Automatic Graph Clustering. In S. North, editor, Graph Drawing, Lecture Notes in Computer Science, Springer Verlag, 1996.

[15] T. Roxborough and Arunabha. Graph Clustering using Multiway Ratio Cut. In S. North, editor, Graph Drawing, Lecture Notes in Computer Science, Springer Verlag, 1996.

[16] B. Stein and O. Niggemann. On the nature of Structure and ist Identification. In 25. Workshop on Graph Theory, Lecture Notes in Computer Science, Springer Verlag, 1999.

[17] B. Huffaker, E. Nemeth, k claffy. Otter: A general-purpose network visualization tool. http://www.caida.org/Tools/Otter/Paper

[18] B. Huffaker, J. Jung, D. Wessels, k claffy. Visualization of the Growth and Topology of the NLANR Caching Hierarchy. http://www.caida.org/Tools/Plankton/Paper

[19] P. Eades. A Heuristic for Graph-Drawing, Congressus Numerantium, Vol 42, pp 149-160, 1984.

[20] T. Kamada and S. Kawai, An algorithm for Drawing General Undirected Graphs, Information Processing Letters, Vol 31, pp 7-15, 1989.

[21] T. Fruchterman and E. Reingold, Graph-Drawing by Force-Directed Placement, Software-Practice and Experience, Vol 21, pp 1129-1164, 1991.

[22] K. Sugiyama, S. Tagawa and M. Toda, Methods for Visual Understanding of Hierarchical System Structures, IEEE Transactions on Systems, Man, and Cybernectics, Vol 11, 1981.

[23] R. Beale and T. Jackson, Neural Computing, an introduction, Institute of Physical Publishing, Bristol and Philadelphia, 1994.

[24] R. H. Myers, Classical and Modern Regression with Applications, Duxbury Press, Boston, 1986.

[25] D. W. Hosmer and S. Lemeshow, Applied Logistic Regression, John Wiley & Sons, New York, 1989.