Network Security Using Graph Theory

[1]Rogith.A, [2]Sangeetha.R, [3]Nivethitha.R
[1]Msc. Mathematics, [2]Msc. Mathematics, [3]Msc. Mathematics
[1]Sri krishna arts and science college

Abstract **- network monitoring is the foremost obligation for any network security. In this paper, for monitoring network activities we present concept of traffic dispersion graph which can help easy identification of access pattern over a network. We define adjacency matrix attack graph to examine and detect possible risks to safeguard the critical network system against multi step attacks. Network optimization can create the effect of increasing capacity of the network. Prioritizing traffic is a highly efficient way to enhance network performance and we can better share the available resources by allowing intermediate node to perform the operations, in addition to forwarding packets.**

keywords **- network monitoring, traffic dispersion graphs, attack graphs, network coding, vulnerability.**

I. INTRODUCTION

With the development in global mobile communication networks there have also been various problems with it, which needs new solutions. Through the course of the paper, we shall be examining the computer networks, the flow of traffic or data from one computer to another and also the use of techniques of network coding to improve several aspects of network monitoring. Lets assume every computer to be a node in a graph. we consider here the computers as the edges and the data flows in between these edges.

The structure of the paper is as follows., Section 2 is about the concept of traffic dispersion graphs, Section 3 is about attack graphs, Section 4 is about network coding techniques, Section 5 concludes the paper.

II. TRAFFIC DISPERSION GRAPHS

A graphical representation of interaction of various group of nodes is called as a traffic dispersion graph. Generally, the directed edges in a TDG can be used to identify the initiator of the interaction between a pair of nodes. One of the major problem in mobile communication networks is network traffic i.e., amount of data moving across a network at a given point of time. Network packets aggregate to form network data, which provide the load in the network. Here we use the traffic dispersion graphs as a module to observe, examine, envision the occurring traffic network. With the TDG model we can clearly observe the data packets passed from the sender to the receiver and thus we will know the direction in which the data packets have been passed between the nodes of the edges. And then comes the process of edge filtering i.e., the level of interaction between the pair of nodes. This works like no extra data packets are allowed to translate after an edge is added by the filter in between the two nodes. A graph is a collection of group of hosts, host is where the flow of packets take place and an edge can represent the exchange of at least one packet and thus TDGs can be seen as the natural next step in the progression of packet, flow, and host level aggregation. Here we propose TDGs as an alternative way to avoid network traffic, i.e., TDGs are different from other sort of applications which have a different structure and visualizing methods.

TDG FORMATION

In this paper we use port based TDGs i.e., numbers used to uniquely identify a transaction over a network by specifying both the host and the service. So, for different ports we use different types of edge filters. For example, for transmission control protocol (TCP) we use EFSP edge filters on the corresponding destination port and for user datagram protocol (UDP) interaction we use EFP edge filter on the destination port of interest. Here we will be referring to the port based TDGs which uses the name of the dominant of that port. For example, the HTTP TDGs is formed when a HTTP client initiates a request by establishing a TCP connection to a particular port on a server and the HTTP port number is 80. Firstly, we will be using the edge filtering process so that the TDGs capture any applications that is used by these ports.as because there are also so-called non-standard applications, which uses standard ports like HTTP. And we use this port-based filtering only because it is consistent i.e., if at any point the traffic at TCP port 80 appears significantly different, it could be like some other non-related application tunnels its traffic in that port or there may be a behavioral change of the application.

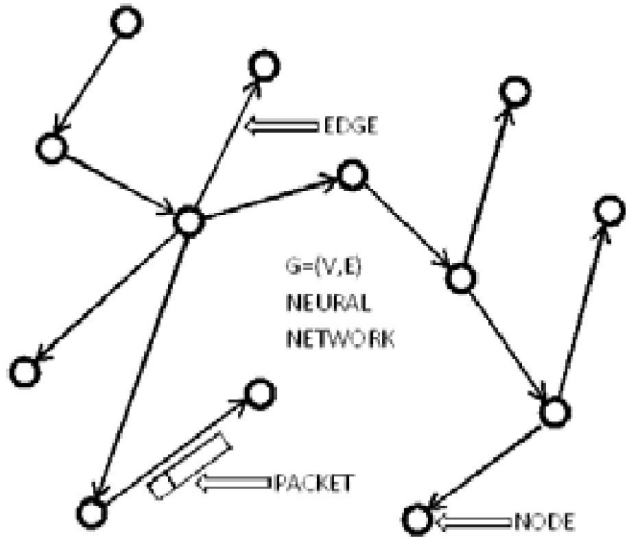

Figure 1 Traffic Dispersion Graph

TDG VISUALIZATION

TDG is a novel way to analyze network traffic with a powerful visualization. Usually where in IP networks, it takes place like a node of the TDG corresponds to an entity with a distinct IP address and the exchange of packets between various sender and destination nodes is captured by the graph. In figure 1 we can see that TDG s have significant visual and structures. TDGs capture many interesting patterns of node interactions. The node degrees help in visually determining the type of relationship between the nodes. The role of the node can be inferred from the direction of the edges. There also exists node chains. The nodes have their own communities and their sizes. Our goal here is to visualize the properties of the nodes.

III. ATTACK GRAPHS

ATTACK GRAPHS GENERATION

Attack graphs are data structures that model all possible avenues of attacking a network. There are several tools which are used to measure the point-based vulnerabilities on individuals hosts. However, vulnerabilities on a network being of casual relationships actually arouse more impact and damage to a whole network and persist longer and more undetectable if we are unable to defend against them in relevance. Attack graphs of automated generation encode the casual relationships among vulnerabilities and tell whether critical assets are secure against potential multi-step combining attacks. A piece of software which enables people to define software testing tasks are the automation tools which are used to automate the generation of attack graphs and releases administrators from error-phone and arduous manual work. Therefore, it has become a desirable tool for administrators to analyze their networks, report potential risks and protect their networked assets. But there is a limitation to it, the complexity problem, regarding the size of the network and vulnerabilities that exist in the network. Practically, attack graphs always have more ability than human to visualize, and understand.

EXAMPLE: A network configuration shows connections between machines and vulnerabilities distribution on a network. The dependency between vulnerabilities is done by the type of graphs or it may also exploit the relations between vulnerabilities. Out of the two inputs, a vulnerability-based attack graph can be drawn out, in which a security-related vulnerability or condition represents the system state, and an exploit between vulnerabilities is modeled as a transition. Figure illustrates a network configuration example.

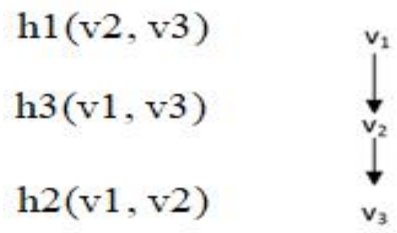

Fig 2 Dependencies

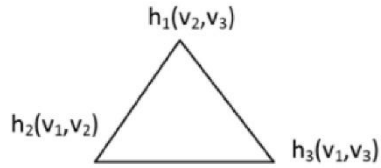

Fig 3 Machines

Having vulnerabilities v2 and v3.h2 has vulnerabilities v1 and v2.h3 has vulnerabilities v1 and v3. The type of graph on the right side expresses the dependent relations between vulnerabilities. v1 is the first vulnerability that is assumed satisfied on its own. v2 is dependent on the satisfaction of v1.v3 is dependent on the satisfaction of v2. Therefore, v1 is the precondition of v2; v2 is the precondition v3. In another way, we can say v2 is the post condition of v1 and v3 post condition of v2. Acquiring the vulnerability-based attack graph has many approaches. A primary method to acquire a vulnerability-based attack graphs is to find all the attack paths, and then use them to set up an attack graph.

$h_2v_1 \rightarrow h_1v_2 \rightarrow h_1v_3$	$h_3v_1 \rightarrow h_1v_2 \rightarrow h_1v_3$
$h_2v_1 \rightarrow h_1v_2 \rightarrow h_3v_3$	$h_3v_1 \rightarrow h_1v_2 \rightarrow h_3v_3$
$h_2v_1 \rightarrow h_2v_2 \rightarrow h_1v_3$	$h_3v_1 \rightarrow h_2v_2 \rightarrow h_1v_3$
$h_2v_1 \rightarrow h_2v_2 \rightarrow h_3v_3$	$h_3v_1 \rightarrow h_2v_2 \rightarrow h_3v_3$

Fig 4 Attack Path

ADJACENCY MATRIX CLUSTERING

The rows and columns of an adjacency matrix could be placed in any order, without affecting the structure of the attack graph the matrix represents. In a graph structure the orderings capturing the regularities are clearly desirable. To point out, we need orderings that combine graph vertices (adjacency matrix rows and columns) by common edges. With matrix clustering we can cluster of common edges as a single unit as we analyze the attack graph. There are some cases, where the network attributes allow us to order adjacency matrix rows and columns into clusters of common attack graph edges i.e., the machines in the same subnet will appear in consecutive rows and columns of the adjacency matrix by accordingly sorting the IP address and the machine vertices. connected blocks of elements on the main diagonal can be caused by unrestricted connectivity within each subnet.

Generally relying on a priori ordering of rows and columns to place the adjacency matrix into meaningful clusters cannot be done. We therefore apply a particular matrix clustering algorithm that is designed to form homogeneous rectangular blocks of matrix elements. When there are similar patterns of attack graphs within a block we call it homogeneous. This clustering algorithm requires no user intervention, has no parameters that need turning and scales linearly with problem size. With this process of clustering, the number of rows and columns to the cluster are assigned where they form the region of high and low densities. The cluster optimality is obtained by the number of clusters and cluster assignments. The matrix clustering algorithm is based on ideas from data compression, including the minimum description length principle, in which regularity in the data can be used to compress optimality. When we compress the data more we will have better captured regularities so that can understand it in a better way ahead.

m_{v_1}			m_{v_2}			m_{v_3}		
0	0	0	$h_1h_1v_2$	$h_1h_2v_2$	0	$h_1h_1v_3$	0	$h_1h_3v_3$
0	h_2v_1	0	$h_2h_1v_2$	$h_2h_2v_2$	0	$h_2h_1v_3$	0	$h_2h_3v_3$
0	0	h_3v_1	$h_3h_1v_2$	$h_3h_2v_2$	0	$h_3h_1v_3$	0	$h_3h_3v_3$

Fig 5 Adjacency Matrix Clustering

MATRIX OPERATIONS FOR MULTI STEP ATTACKS

The adjacency matrix shows the presence of each edge in a network attack graph. Every certain single-step attack is deliberately shown by the adjacency matrix i.e., the adjacency matrix shows attacker reachability within one attack step. Whereas for multi-step attacks by repetitious method of matching rows and columns we can navigate the adjacency matrix. Adjacency matrix can be raised to higher powers. For a square(n*n) adjacency matrix A and a positive integer p, then A^p is raised to the power p: in other words,

$$A^p = (A\ AA.\ A) \text{ p times} \qquad (1)$$

Here, the matrix multiplication is in the usual sense. For example, an element of A^2

The matrix multiplication where the matching of rows and columns is corresponding to the attack graphs matching steps, thus by the equation. The number of matching steps is summed by k. Accordingly, the 2-step, 3-step attacks and the 4-step attacks are given by A^2, A^3 and A^4.

For raising a (square) matrix to an arbitrary power, we can improve upon iterative multiplication. This involves a spectral decomposition of A. An n*n matrix always has n eigen values. A diagonal matrix is formed in a order of n*n and a corresponding matrix of nonzero columns V that satisfies the eigen value equation AV=VD. V is invertible if the n eigen values are distinct so we can write as,

$$A=VDV^{-1} \quad (2)$$

It is then straightforward to prove that $A^p=VD^pV^{-1}$, via $V^{-1}V=1$. D^p is the diagonal matrix of power p. Thus, the product VD^pV^{-1} is computed easily i.e.,

$$D^p = \begin{bmatrix} d_1^p & 0 & \cdots & 0 \\ 0 & d_2^p & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & d_n^p \end{bmatrix}.$$

Fig 6 Diagonal Matrix from Eigen Value

ATTACK PREDICTION

In our approach, we place detected intrusions within the context of predictive attack graphs based on known vulnerability paths. From our knowledge of network configurations, attacker exploits we compute a vulnerability-based attack graphs followed by forming A which is a adjacency matrix for performing algorithms such as clustering to accomplish security in networks and we start computing the matrix.

So with the generation of the alarm, we can associate it with the following elements.

- The adjacency matrix A (for single-step reachability).
- The transitive closure of A. On the basis of number of associated attacks steps we can instantly cover up the alerts. To define it, we know that according to attack graphs it is impossible for an alarm to occur in a zero-valued region, so it is considered as a false alarm. So as with the help of alarms which occurs, we will know about the type of attacks taking place i.e., when it occurs within a single-step of the matrix then it is a single-step attack in the attack graph. And when it occurs in any p-step region then it takes of about p-steps to achieve that attack. We can also predict the origin and the impact of attacks with the association of intrusion alarms and the graph where it occurs. So in one of the vulnerability -based reachability graph when placed an intrusion alarm we can detect the type of attacks.
- When the row and column indices are equal the idea is to project to the main diagonal of the graph. Vertical projection leads to attack step in forward direction i.e., attack steps are in forward direction when one projection is made from the column to the main diagonal and it is given by the resulting row of the column projection. The prediction of attacks is made in several ways such as one step away, multiple steps away and all the steps combined. Which is explained below
- The projection can be done repeatedly one at a time in a attack.
- Similarly, projection can be done repeatedly choosing single-step elements or multi-step elements and in multi-step way the projected row shows the minimum number of steps needed to reach another matrix.
- Transitive closure is used here when all the steps are combined as it gives reachability over all steps there is no need of iteration of the projection. The impact of the graph is predicted here by the projection along the column of the matrix. Correspondingly, we can project along a row of such a matrix to predict attack origin. Where the projection is done along a row to the main diagonal, the resulting column gives the possible steps backward in the attack. As there are three ways for the forward projection the same is for the backward direction.

IV. CONCLUSION

Thus, the rapid growth in global mobile communication networks demands new solutions for existing problems. So, the challenges can be overcome by application of graph theory as TDGs, attack graphs, network coding etc., in wireless mobile networks.

REFERENCES

[1] Bondy, J. A.; Murty, U. S. R. (2008). Graph Theory. Springer.
[2] Newman, Mark (2010). Networks: An Introduction. Oxford University Press.
[3] Mojo H., (2016). Graph theory. Available at www.en.m.wikipedia.org/wiki/Graph_Theoy. Retrieved 9th August 2016.
[4] Deo, Narsingh (1974). Graph Theory with Applications to Engineering and Computer Science, Englewood, New Jersey: Prentice-Hall.
[5] Sumathy et. al.," Secure key exchange and encryption mechanism for group communication in wireless ad hoc networks", Journal on Applications of graph theory in wireless ad hoc networks and sensor networks, Vol 2, No 1, March 2010.
[6] Oliveira, L.B., Wong, H.C., Dahab, R., Loureiro, A.A.F.: On the design of secure protocols for hierarchical sensor networks. International Journal of Security and Networks (IJSN) 2(3/4) (2007) 216–227 Special Issue on cryptography in Networks.
[7] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based net- work vulnerability analysis. In Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02), 2002.
[8] Budayasa, I. K.., Teori Graf dan Aplikasinya (Graph Theory and Its Applications) Surabaya: University Press of Universitas Negeri Surabaya, 2007.