Subscription Agency
Exploring and Distributing Knowledge Globally

Innovative Journal of Computer Technology
Volume 2 Issue 1; 2022

Graph Based Models in Network Security

P.Prasanya Devi[1*], S.Kannan[2]
[1]Research Scholar, [2]Professor, Department of Computer Application, School of Information Technology, Madurai Kamaraj University, Madurai
Corresponding Author
Email Id: prasanyamsc@gmail.com; skannanmku@gmail.com

ABSTRACT

A graphical representation is generated to provide an impression about the complexity of the reviewed system. The presented work is part of an ongoing doctoral thesis and is therefore at an early research stage. The approach is explained by a practical example. The perspective directions in evaluating network security are simulating possible malefactor's actions, building the representation of these actions as attack graphs (trees, nets), the subsequent checking of various properties of these graphs, and determining security metrics which can explain possible ways to increase security level. The paper suggests a new approach to security evaluation based on comprehensive simulation of malefactor's actions, construction of attack graphs and computation of different security metrics. The approach is intended for using both at design and exploitation stages of computer networks.

Keywords: *Network Security, Connected graph, Attack graph, Malware identification, Traffic Dispersion Graph, Network Monitoring.*

INTRODUCTION

A graph based tool can identify the set of attack paths that have a high probability of success for the attacker. We may have a question, why we use the concept of graph theory for the risk in the process of communication from an IP address on a vulnerable port? The reason is, graph has been used to discuss relationships between nodes and the flow of traffic on the communication path which is easy to understand. It is also easy to find the vulnerability and the avoidance of vulnerability in communication network.

The increase of networks and security mechanisms complexity, vulnerabilities and potential operation errors as well as malefactors' possibilities causes the necessity to develop and use powerful automated security analysis techniques. These techniques should allow revealing possible assault actions, determining vulnerabilities, critical network resources and security bottlenecks, and finding out and correcting errors in network configurations and security policies.

Data network consists of a huge amount of different traffic sources that make difficulties to find abnormal and unwanted traffic sources between them. Available signature based detectors are able to identify only well known threads and to apply predefined actions to prevent them. Advanced experience based systems, for instance case-based reasoning, depends on similar case retrieval

The objective of this paper is to review graph based similarity method for describing individual host role in the network, based on their relationships with other hosts. The method relays only on the fact that there is communication between pair of hosts, without taking into

account unreliable information about payload, port numbers, protocols, etc. We will analyze single and multi sensor traffic flow capture aspects. This work is focused on the TCP/IP network protocols. This paper presents a graph based approach to network vulnerability analysis. The method is flexible, allowing analysis of attacks from both outside and inside the network. It can analyze risks to a specific network asset, or examine the universe of possible consequences following a successful attack. The analysis system requires as input a database of common attacks, broken into atomic steps, specific network configuration and topology information, and an attacker profile. The attack information is matched with the network configuration information and an attacker profile to create a superset attack graph. Nodes identify a stage of attack, for example the class of machines the attacker has accessed and the user privilege level he or she has compromised. The arcs in the attack graph represent attacks or stages of attacks. By assigning probabilities of success on the arcs or costs representing level of effort for the attacker, various graph algorithms such as shortest path algorithms can identify the attack paths with the highest probability of success.

RELATED WORK

There is a significant body of literature in the area of deploying measurement points and studying their characteristics. However, only few projects focused on minimizing the overhead of such deployment IDMaps [1] studies distance monitoring and estimation by finding distance between Tracers, which are monitoring boxes, placed at various network nodes. The distance maps form the virtual topology of the Internet. However, IDMaps does not assume that each tracer monitors a shortest path tree as it is assumed here. [2] and [3] study link monitoring and delays in IP networks based on a single point-of-control. PingTV [4] and Atlas uses ICMP to generate a logical map of the network from a single probing host. PingTV monitors the traffic condition and network outages of networks with hierarchical structure by pinging hosts in hierarchical order. Atlast captures the topology of IPv6 networks by probing from an initial set of seeds that grows whenever new routers discovered. Either of these methods is proactive. That is, it sends network probes whereas in our methods we do not require any network probes. In [5] and [6], the authors develop techniques to infer the performance of all of the links (or specified subset of links) that are contained by the trees. [7] was the first study to show that the concept of ”more measurement points is better” is not accurate by showing that the topology can obtained using few measurement points.

[8] studies deploying minimum number of beacons on a network of known topology and BGP-like routing policy so that every link is monitored by messages originating from at least one beacon. In [9], the authors propose a model to minimize the overhead of monitoring all links of a given network. However, this approach is different from ours as it considers weighted networks and shortest path trees rooted at these nodes are fixed.

CONNECTIONS GRAPH

There exist three categories of case representation [10]: feature value representations, sequences and strings, and structural representations. The first and the second category were observed in my previous publications. The structural representation itself can be divided in three categories: hierarchical structure, network structure, and flow structure. The most straightforward structure representing network communications is the networkstructure.

In the field of similarity findings for networks and graphs there exist several well known methods, like structure mapping engine (SME), graph edit distance, largest common subgraph. All these methods are computationally expensive, thus they are not applicable ina

huge structures like communication graphs in real networks. Still graph theory is successfully applied to solvedifferent tasks in telecommunication networks. Dense bipartite graph are used for spam filtering. Virus propagation and its epidemical threshold also are obtained from the graph theory. Well known graph is global routing topology of the Internet.

In general the graph *G* (*V, E)* consists of a set of vertices V and a set of edges E. An edge e_{ij} represents the connection between vertex i and j (unique IP addresses).The network is dynamic environment, where we need to fix some time windows to measure similarity. The time window can be fixed or sliding. There are two general methods how to construct graph from network traffic [11] - the first is Edge on First Packet (EFP) filter that characterizes protocol independency. The second is Edge on First SYN Packet (EFPS) that is more accurate, but is applicable only for TCP flows, therefore not usable in more general cases like this.

The source of data in my research is netflow data [8] obtained from routers. As it follows from TCP/IP model, the routers operate at internetwork level and the communication inside broadcast domain does not traverse them (Fig. 1) – thus any local communication F5, F6 are not captured.

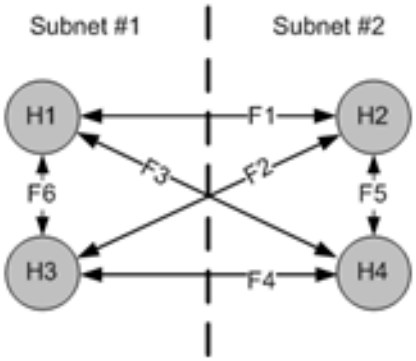


Figure 1.Subnet boundary and graph limitations

EXISTING SOFTWARE FOR GENERATING NETWORK GRAPHS

A network is a graph G = (V,E), where V is the set of nodes and E is the set of direct communication lines between nodes, called edges. The number of nodes and edges are respectively denoted by |V | and |E|. Observe that in an Internet Service Provider (ISP) network consisting of a single Open Shortest Path Protocol (OSPF) area, each node v in the network forms a shortest path routing tree to route to all other nodes in the network. A shortest path between nodes s and t is denoted by Ps,t and the length *(i.e.* number of edges) in this path is denoted by d(s, t). Clearly, between any two nodes there exists possibly more than one shortest path. However, as the name suggests all these paths have the same length d(s, t). For the purposes of our discussion we assume that graph G representing the network is connected. That is, there is a path between any two nodes in V. Consider node v. A shortest path tree Tv rooted at v is a spanning tree of G which, for any node vi, contains a shortest path between v and vi. We assume that the graph is unweighted. That is, all links (i.e. edges) in the network have the same weight. This can occur, for instance, when OC-48 connections are used as the ISP backbone, all the links have the same capacity, or all the links have delay below an appropriate threshold. Let G be a graph and $v \in V$. We call an edge $e = (a, b) \in E$

horizontal with respect to v if d(v, a) = d(v, b) in G. The following observation states that a SP-tree rooted at a node v cannot cover any edge that is horizontal with respect to v.

Observation 1

Let G = (V,E) be a graph and v be a node of G. Any edge of G which is horizontal with respect to v cannot belong to any SP-tree Tv rooted at v.

Proof: Suppose that e = (a, b) is a horizontal edge with respect to v but nevertheless there is a SP-tree Tv containing e. Since the shortest distances from v to a and b are the same, we obtain that there is a loop in Tv, which is a contradiction.

We call an edge e = (a, b) ∈ E vertical with respect to v if d(v, a) = d(v, b) + 1 or d(v, b) = d(v, a) + 1. Let e = (a, b) be a vertical edge with respect to v and assume that d(v, a) = d(v, b)+1 holds. Then, e is called an unavoidable edge by v if any shortest path between v and a includes Observe that if edge e = (a, b) is unavoidable with respect to v, then e = (a, b) is also vertical edge with respect to v. However, the opposite is not necessarily correct. To illustrate, in the network depicted on Fig. 1, edge (2, 3) is unavoidable and edge (2√ n+2) is vertical (but not unavoidable) with respect to node 1. The following observation holds.

Observation 2

Let G = (V,E) be a graph and v be a node of G. Let Sv be the set of all possible shortest path trees rooted at node v. Edge e of G is unavoidable by v if and only if any tree Tv ∈ Sv contains e.

Proof: Clearly, if any tree Tv ∈ Sv contains e = (a, b), then e is vertical with respect to v and any shortest path from a to v includes b if d(v, a) = d(v, b) + 1. Suppose now, that there is a tree Tv in Sv that does not contain e = (a, b) and e is an unavoidable edge by v. Assume without loss of generality that d(v, a) = d(v, b) + 1. Since Tv is a SP-tree, every node of G must be a node of the tree.Consequently, both a and b are in Tv but the edge (a, b) is not in Tv. On the other hand, since (a, b) is unavoidable by v, any shortest path from a to v contains node b. A contradiction obtained completes the proof.

We are interested in the following three problems and describe their applications in practice

- E-Problem (”Exist”-Problem): Given a graph G = (V,E), select a minimum set of nodes R ⊆ V and for each node v ∈ R a tree Tv ∈ Sv such that the union of selected trees covers all edges of G.
- A-Problem (”Any”-Problem): Given a graph G = (V,E), select a minimum set of nodes R ⊆ V such that, regardless which tree Tv ∈ Sv is selected for a node v ∈ R, the union of those trees cover all edges of G.
- BR-Problem (”Bejerano/Rastogi”-Problem): Given a graph G = (V,E) and a SP-tree Tv for each v ∈ V , select a minimum set of nodes R ⊆ V such that the union of Tv, v ∈ R, covers all edges of G.

First we demonstrate that optimal solutions for each of the first two problems are considerably different. (Clearly, the size of the optimal solution to the third problem depends on the choice of the family {Tv : v ∈ V }.) Consider, for example a rectilinear grid of size √ n × √n depicted on Fig. 1. We number nodes of the grid from 1 to n row-wise (ith row nodes are (i−1) √ n+1, (i−1) √ n+2, . . . , i√n).

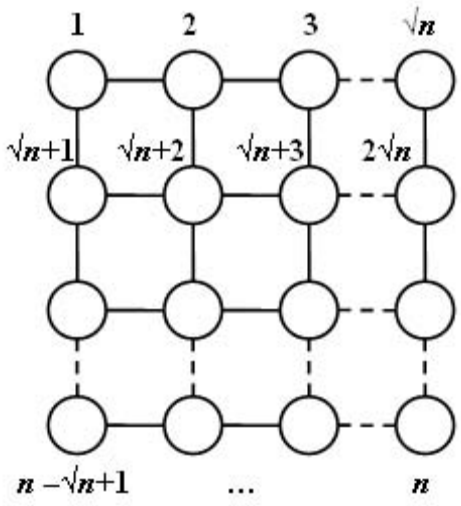


Figure 1. A-Problem √ vs E-Problem. On the rectilinear grid of size n ×√n, optimal solution to the E-Problem consists of two trees while optimal solution to the A-Problem consists of √n trees.

One can select two SP-trees, one rooted at node 1 and another rooted at node n, such that their union covers all the edges of G. Indeed, one can consider a tree rooted at node 1 which is formed by edges ((i−1) √n+ 1, i√n+1), (j, j + 1) and (I√n + j, i √n + j + 1) (so called ”first column and all rows”-tree) and a tree rooted at node n which is formed by edges (n−√n+j, n−√n+j +1),(i√n, (i+1)√n) and ((i−1)√n+j, i√n+j) (so called ”last row and all columns”-tree), where 1 ≤ i ≤√n−1,1 ≤ j ≤√n − 1. It is easy to see that both trees are SP-trees. Thus, a solution to the E-problem consists of only two SP-trees. In view of Observation 2, it is rather simple to show also that an optimal solution to the A problem consists of √n roots (SP-trees). Indeed, since edges of column j (j = 1, . . . ,√n) are unavoidable only by nodes of this column, at least one SP-tree rooted at anode of column j must be present in an optimal solution. Any SP-tree rooted at this node will cover all edges of column j, and for any node x not from column j and any edge e from this column, there exists a SP-tree rooted at x which does not contain the edge e. Analogously, at least one SP-tree rooted at a node of row i (i = 1, . . . , √n) must be present in an optimal solution. Thus, an optimal solution to the A-problem on a rectilinear grid of size √n×√n must consist of √n SP-trees with roots one per each column and each row. It is easy to see that selecting the nodes on a diagonal of the grid generates a set of SP-trees that completely cover all edges of the grid.

WAXMAN MODEL

We generate 300-node network topologies using the Wax-man model, which is a popular topology model for networking research. Different network topologies can be generated by varying three parameters:

1) n, the number of nodes in the network graph
2) α, a parameter that controls the density of short edges in the network;
3) β , a parameter thatcontrols the average node degree.

For our experimental purposes, we start with the values α=0.15 and β=0.2 (i.e. average degree = 6), we fix the value of α and increase the value of β gradually to simulate dense network with average low degree. Then, we repeat the same process but by fixing the value of β and increasing the value of α to simulate networks with short links and average high

degree (i.e. average degree = 59). The reason for increasing values of α and β instead of keeping them fixed is that real ISP topology maps can be completely different from each other as shown in Figure 9. Figure 7 shows the results we got running the heuristic of the A-problem as well as differ- ent heuristics for the E-problem on networks generated using Waxman model.

POWER-LAW MODEL

Power-Law model can be used to generate router-level topologies [3]. For our experimental purposes, we generate 300-node flat (i.e. non hierarchical) power-law topologies using BRITE [5], which is the best known power-law-based topology generator. BRITE generates different topologies by changing the values of the following parameters: (1) *HS*, Size of one side of the plane; (2) *LS*, Size of one side of a high-level square; (3) *NP* , Node Placement; (4) *m*, Number of links added per new node; and (5) *IG*, Incremental Growth.

We start with average degree m=2 (i.e. tree topology) and increase the degree gradually until m = 10 (i.e. average degreeof 10). Figure 7 shows the results we got running the heuristics of the A-problem as well as E-problem on networks generatedusing Power-Law model.

NETWORK SIMUATOR MODEL

Network Simulator (ns) is a discrete event simulator targeted at networking research. Ns provides substantial support for simulating communication protocols such as TCP, routing protocols, and multicast protocols over wired and wireless (both local and satellite) networks. Ns have the capability of simulating live networks by introducing live traffic. This is done by injecting traffic from the simulator into the live network.

The simulator acts like a router allowing real-world traffic to be passed through without being manipulated. The ns packet contains a pointer to the network packet. Network packets may be dropped, delayed, re- ordered or duplicated by the simulator. Opaque mode is useful in evaluating the behavior of real-world implementations when subjected to adverse network conditions that are not protocol specific.

The real-time scheduler implements a soft real-time scheduler which ties event execution within the simulator to real time. Provided sufficient CPU horsepower is available to keep up with arriving packets, the simulator virtual time should closely track real-time. If the simulator becomes too slow to keep up with elapsing real time, a warning is continually produced if the skew exceeds a pre-specified constant ``slop factor''.

TRAFFIC DISPERSION GRAPHS

A major problem these days is keeping a check on the traffic and thus detecting applications that are not required. This is because many applications obfuscate their traffic using unregistered port numbers or payload encryption. In this paper, we propose the use of Traffic Dispersion Graphs (TDGs) as a way to monitor, analyze, and visualize network traffic. TDGs model the social behavior of hosts ("who talks to whom"), where the edges can be defined to represent different interactions (e.g. the exchange of a certain number or type of packets). With the introduction of TDGs, we are able to harness a wealth of tools and graph modeling techniques from a diverse set of disciplines. In this work, we propose a different way of looking at network traffic that focuses on network-wide interactions of hosts (as seen at a router. We argue that there is a wealth of information embedded in a TDG. For example,

a popular website will have a large in-degree, while P2P hosts will be tightly connected. An edge can represent the exchange of at least one packet. In other words, a TDG can represent a particular type of interaction, which gives them significant descriptive power, as we discuss later in detail. TDGs can be seen as the natural next step in the progression of packet, flow, and host level aggregation. This is because flow aggregates a set of packets, a host aggregates a set of flows originating and terminating at the host and a graph aggregates a group of hosts. Our main goal is to propose TDGs as a different way of modeling traffic behavior, and show that they: (a) have characteristic structure and provide visualizations that can distinguish the nature of some applications, (b) describe traffic along a new "dimension", the network-wide social behavior, which complements traffic characterization at the packet, flow and host levels.

TDG FORMATION

In this paper we focus on port-based TDGs. Throughout the paper and unless stated otherwise, when the legacy application for a port uses TCP, we use the EFSP edge filter on the corresponding destination port (*e.g.*, TCP Port 25 for SMTP). When we examine UDP interactions, we use the EFP edge filter on the destination port of interest (e.g., UDP Port 53 for DNS). For ease of presentation, we will refer to each port-based TDG using the name of the dominant or well-known application under that port. For example, the HTTP TDGs is formed by using as edges all the TCP SYN packets that have as destination port the number 80. Since we use edge filtering by port number, the TDGs capture aspects of any application that uses these ports. We are fully aware that many non- standard applications, such as P2P traffic use standard ports such as Port 80. However, port-based filtering is consistent with our use of TDGs as a monitoring tool. For example, if at some point traffic at TCP Port 80 appears significantly different, it could be:

a) a new benign or malicious application tunneling its traffic under that port, or
b) a change in the behavior of the traditional application.

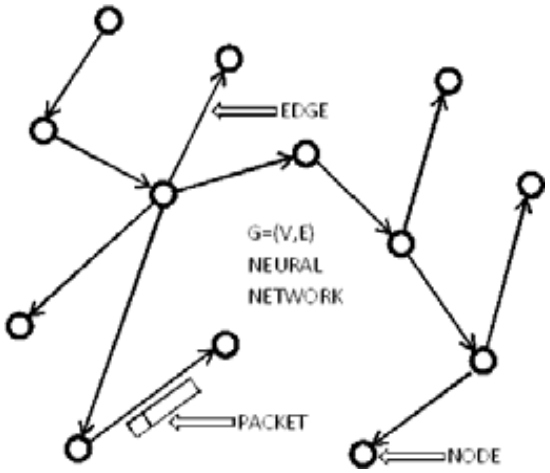


Fig. 2 Traffic Dispersion Graph

TDG VISUALIZATION

Traditionally, visualization of traffic in monitoring tools has largely been limited to visualizing measures of traffic volumes on a per flow basis. By contrast, we show that TDGs lend themselves to simple graphical visualizations of interaction patterns. We can identify several distinctive structures and patterns in TDGs, which are indicative of the behavior of

different applications. Node degrees - The degrees of various nodes and their connectivity in a TDG helps us in visually determining the type of relationship between the nodes.

ATTACK GRAPHS

Attack Graphs Generation

Several tools measure point-based vulnerabilities on individual hosts. However, vulnerabilities on a network being of causal 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 causal relationships among vulnerabilities and tell whether critical assets are secure enough against potential multi-step combining attacks. The automated tool succeeds to automate the generation of attack graphs and releases administrators from error-prone 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, that is, the complexity problem, regarding the size of the network and vulnerabilities that exist in the network. In practice, attack graphs always exceed human ability to visualize, and understand.

Motivating Example

A network configuration shows connections between machines and vulnerabilities' distribution on a network. A type graph tells the dependency or exploits 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. The left side is the network configuration graph. h1 is a machine

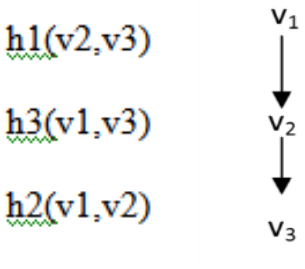


Fig. 2: Dependencies

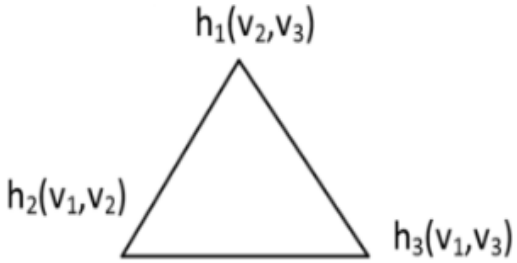


Fig. 3: Machine

Having vulnerabilities v2 and v3 (these vulnerabilities are generalized with simplified notations, which do not express any concrete vulnerability but conceptual ones mainly for their relationships). h2 has vulnerabilities v1 and v2. h3 has vulnerabilities v1 and v3. The right side is a type graph that expresses 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 pre-condition of v2; v2 is the precondition of v3. In another way, we can say v2 is the post-condition of v1 and v3 is the post-condition of v2. Here the satisfied or satisfaction means that vulnerability on a machine, whose preconditions have all been satisfied by an attacker, can be reached or acquired by the attacker now. Acquiring the vulnerability-based attack graph has many approaches. However, a direct way is to find all the attack paths, and then uses 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 Paths

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. But orderings that capture regularities in graph structure are clearly desirable. In particular, we seek orderings that tend to cluster graph vertices (adjacency matrix rows and columns) by common edges (non-zero matrix elements).

This allows us to treat such clusters of common edges as a single unit as we analyze the attack graph (adjacency matrix). In some cases, there might be network attributes that allow us to order adjacency matrix rows and columns into clusters of common attack graph edges. For example, we might sort machine vertices according to IP address, so that machines in the same subnet appear in consecutive rows and columns of the adjacency matrix. Unrestricted connectivity within each subnet might then cause fully connected (all ones) blocks of elements on the main diagonal.

In general, we cannot rely on a priori ordering of rows and columns to place the adjacency matrix into meaningful clusters. We therefore apply a particular matrix clustering algorithm [23] that is designed to form homogeneous rectangular blocks of matrix elements (row and column intersections). Here, homogeneity means that within a block, there is a similar pattern of attack graph edges (adjacency matrix elements). This clustering algorithm requires no user intervention, has no parameters that need tuning, and scales linearly with problem size. This algorithm finds the number of row and column clusters, along with the assignment of rows and columns to those clusters, such that the clusters form regions of high and low densities. Numbers of clusters and cluster assignments provide an information-theoretic measure of cluster optimality. The matrix clustering algorithm is based on ideas from data compression, including the Minimum Description Length principle [40], in which regularity in the data can be used to compress it (describe it in fewer symbols). Intuitively, one can say that the more we compress the data, the better we understand it, in the sense that we have better captured its regularities.

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. Taken directly, the adjacency matrix shows every possible single-step attack. In other words, the adjacency matrix shows attacker reachability within one attack step. As we describe later, we can navigate the adjacency matrix by iteratively matching rows and columns to follow multiple attack steps. We can also raise the adjacency matrix to higher powers, which shows multi-step attacker reachability at a glance. For a square (n × n) adjacency matrix A and a positive integer p, then A^p is A raised to the power p: In other words,

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

Here, matrix multiplication is in the usual sense. For example, an element of A^2. In Equation, the matching of rows and columns in matrix multiplication (index k) corresponds to matching steps of an attack graph. The summation over k counts the numbers of matching steps. Thus, each element of A^2 gives the number of 2-step attacks between the corresponding pair (row and column) of attack graph vertices. Similarly, A^3 gives all 3-step attacks; A^4 gives all 4-step attacks, etc.

For raising a (square) matrix to an arbitrary power, we can improve upon naïve iterative multiplication. This involves a spectral decomposition [41] of A. An n × n matrix always has n Eigen values. These form an n × n diagonal matrix D and a corresponding matrix of nonzero columns V that satisfies the Eigen value equation AV = VD. If the n Eigen values are distinct, then V is invertible, so that we can decompose the original matrix A as

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

Here D is a diagonal matrix formed from the eigen values of A, and the columns of V are the corresponding eigenvectors of V.

It is then straightforward to prove that $A^p = VD^pV^{-1}$, via $V^{-1}V = I$. This product VD^pV^{-1} is easy to compute since D^p is just the diagonal matrix with entries equal to the p th power of those of D, 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 values

Attack Prediction

In our approach, we place detected intrusions within the context of predictive attack graphs based on known vulnerability paths. We first compute a vulnerability-based attack graph from knowledge of the network configuration, attacker exploits, etc. We then form the adjacency matrix A for the attack graph, perform clustering on A. We then compute either the transitive closure of A, or the multi-step reachability matrix.

Then, when an intrusion alarm is generated, if we can associate it with an edge (e.g., exploit) in the attack graph, we can thus associate it with the corresponding element of any of the following:

- The adjacency matrix A (for single-step reachability).
- The multi-step reachability matrix in Equation (6) (for multi-step reachability).
- The transitive closure of A (for all-step reachability).

From this, we can immediately categorize alerts based on the numbers of associated attack steps. For example, if an alarm occurs within a zero-valued region of the transitive closure, we might conclude it is a false alarm, i.e., we know it is not possible according to the attack graph. Or, if an alarm occurs within a single-step region of the reachability matrix, we know that it is indeed one of the single-step attacks in the attack graph. Somewhere in between, if an alarm occurs in a p-step region, we know the attack graph predicts that it takes a minimum of p steps to achieve such an attack. By associating intrusion alarms with a reachability graph, we can also predict the origin and impact of attacks. That is, once we place intrusion alarm on one of the vulnerability-based reachability graphs, we can navigate the graph to do attack prediction.

The idea is to project to the main diagonal of the graph, in which row and column indices are equal. Vertical projection (along a column) leads to attack step(s) in the forward direction. That is, when one project along a column to the main diagonal, the resulting row gives the possible steps forward in the attack. We can predict attack origin and impact either (1) one step away, (2) multiple steps away with the number of steps distinguished, or (3) over all steps combined. Here are those 3 possibilities:

- When using the adjacency matrix A, non-zero elements along the projected row show all possible single steps forward. Projection also can be done iteratively, to follow step-by-step (one at a time) in the attack.

- When using the multi-step reachability matrix in Equation (6), the projected row shows the minimum number of subsequent steps needed to reach another vertex. We can also iteratively project, either choosing single-step elements only, or “skipping” steps by choosing multi-step elements.

- When using the transitive closure, the projected row shows whether a particular vertex can be subsequently reached in any number of steps. Here, iterative projection is not necessary, since transitive closure shows reachability over all steps. We see that projection along a column of a reachability matrix predicts the impact (forward steps) of an attack. Correspondingly, we can project along a row (as opposed to a column) of such a matrix to predict attack origin (backward steps). In this case, when one projects along a row to the main diagonal, the resulting column gives the possible steps backward in the attack. As before, we can predict attack origin using either the adjacency matrix, the multi-step reachability matrix, or the transitive closure matrix. Just as for forward projection, this gives either single-step reachability, multi-step reachability, or all-step reachability, but this time in a backward direction for predicting attack origin.

MALWARE IDENTIFICATION

Once a system has been identified with irregularities from the original work we can predict an malware is trying to attack the system. Computer networks have become a ubiquitous but vulnerable aspect of corporate, university, and government life. Yet the increased complexity of computer networks combined with the ingenuity of attacker’s means that they remain susceptible to expensive attacks from worms, viruses, Trojans, and other malicious

software, which we simply refer to as malware. Network traffic filtering is one of many security methods available tone work administrators. Network traffic filters provide protection by sampling packets or sessions and either comparing their contents to known malware signatures or looking for anomalies likely to be malware. Filtering capabilities have begun to be integrated into routers themselves, so as to reduce hardware deployment costs and to allow for more adaptive security. Future traffic filters are expected to be configurable, networked, and even autonomous. Our objective in this paper is to investigate the deployment and configuration issues of such devices within an optimization framework.

Computer networks have become a ubiquitous but vulnerable aspect of corporate, university, and government life. Yet the increased complexity of computer networks combined with the ingenuity of attackers means that they remain susceptible to expensive attacks from worms, viruses, Trojan's, and other malicious software, which we simply refer to as malware [1]–[3]. Network traffic filtering is one of many security methods available to network administrators. Network traffic filters provide protection by sampling packets or sessions and either comparing their contents to known malware signatures or looking for anomalies likely to be malware. Filtering capabilities have begun to be integrated into routers themselves, so as to reduce hardware deployment costs and to allow for more adaptive security [4]. Future traffic filters are expected to be configurable, networked, and even autonomous. Our objective in this paper is to investigate the deployment and configuration issues of such devices within an optimization framework.

Model and Problem Formulations

There are a lot of goals and restrictions that a Computer network administrator's faces at the network security level and thefinancial or at technical cost of achieving that security level. We combine and express these constraints and objectives within the four malware filter placement problems evaluated in this paper. We consider a network figure, configurable, networked routers with traffic filtering capabilities which can be dynamically and remotely set by a centralized server. Some subsets of these routers are source routers and another (potentially overlapping and typically identical) subset is destination routers. Other routers are core routers. We do not explicitly consider the effectiveness of malware filters. We assume thatfiltered packets are marked so that we do not redundantly filter particular packets. In addition, we assume that the network administrator has full knowledge of the network traffic, possibly with some delay. Finally, we do not consider how the act of filtering malware will alter the quantity of traffic on a link or the quantity of malware at future routers because we assume that the proportion of malware in the network is relatively low. Although we will discuss here packet filtering, all of the developed theory and results also apply to the filtering of sessions.

Centrality Measures for Network Link Assessment

We introduce two new centrality measures within the context of communication networks. Traditional centrality measures, as described in [9], involve source-destination pairs, but each pair is weighted identically. A more relevant and accurate centrality measure would weight source- destination pairs according to the magnitude of traffic that travels between them. Moreover, traditional centrality measures consider every node to be a potential source and destination, but this is not the case for core routers. Therefore, we propose only considering those nodes that are in fact sources and/or destinations (i.e. no core routers) in our centrality calculation. Traffic betweenness centrality (TBC) is betweenness centrality with the above two changes. Let R be a set of all vertices in an undirected graph. Let S⊆ R

contain all the sources, D⊆ R be the set of all destinations, and P be the set of all source-destination pairs (s,d). The number of shortest paths between s א R and d א R is $\imath_{sd}$. The number of these shortest paths that pass through some r א R is $\imath_{sd}(r)$. Moreover, the amount of traffic between s and d is u_{sd}. TBC assumes that there are multiple shortest paths between a source and destination and that they are equally likely to be used. TBC of a router r, denoted by CTB(r), is then defined as the fraction of shortest paths of all source-destination pairs that pass through a particular router, with each source-destination pair being weighted by its traffic magnitude.

$$CTB(r):=X(s,d)אPu_{sd}\imath_{sd}(r)\imath_{sd} \quad (1)$$

If at least one shortest path for any source-destination pair passes through a given router r, then CTB(r) >0. A special case would be based on stress centrality and called traffic stress centrality (TSC). Here we do not assume that all shortest routes are used with equal likelihood but rather that one is chosen. In this case if a node is on this selected shortest route, then $\imath_{sd}$ = 1, otherwise it is 0. TSC of a router r is then defined as :

$$CTS(r) :=X(s,d)אPusd\imath sd(r) \quad (2)$$

There exists an intuitive but not immediately obvious relationship between the traf centrality measures defined here and the actual traffic that passes through a router. Assume that the traffic between all source-destination pairs on this network is routed using either (a) a load-balancing shortest path routing scheme where all the packets sent from a source node to a destination one are equally likely to be delivered through multiple shortest paths between them or (b) a simple shortest path scheme where all packets between the source and the destination nodes are delivered consistently through a single shortest route. Then, the amount of traffic on a router r is equal to the TBC measure in the case of routing scheme (a) and the TSC measure in the case of scheme (b). These relationships can easily be proved by comparing the definitions of traffic centrality measures with simple equations for traffic at routers under these routing schemes. Traffic centrality measures capture the importance of routers on a network, and hence are helpful when defining objective functions for malware filtering problems.

RESULT

The number of packets as the weights between the nodes from one source IP address to destination port number. We calculated the influence from each source IP address to destination port number and identified the most abnormal source IP address to block into the network. From the analysis we found that there is more threat to destination port 1900 from various IP address.

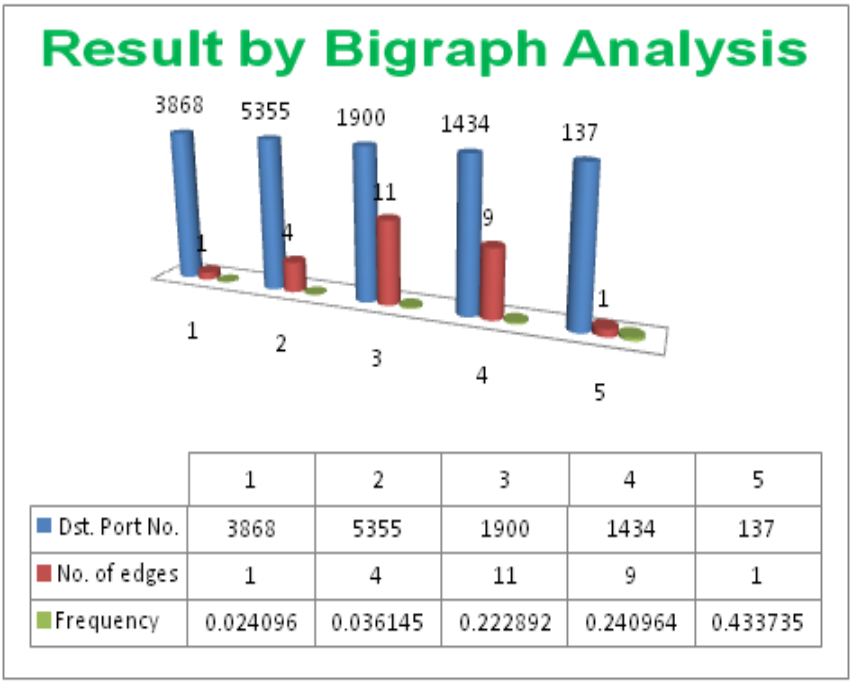
- From the malicious packets obtained, we evaluated, at what frequency these packets influence our network. We considered the following information from the preprocessed data. They are Source IP address, Source Port address, Destination IP address, Destination Port number and the number of packets.
- As per the bipartite graph definition, we consider the two vertices as Source IP address and Destination Port number. Both are disjoint sets and they are also independent sets.
- In our analysis, we separated suspected malicious packets. We identify the malicious or malformed packets based on the following assumptions We found that all malicious packets are using the open ports. We also correlated the edges of the bipartite graph with two assumptions:

Assumption 1: The possible vulnerability is high in a destination port if much source address access them.
Assumption 2: Suspected attacks from a source address is high if its sends more packets to destination ports which are open.

Abnormal Packets Evaluation by Bipartite Graph

From the malicious packets obtained, we evaluated, at what frequency these packets influence our network. We considered the following information from the preprocessed data. They are Source IP address, Source Port address, Destination IP address, Destination Port number and the number of packets.

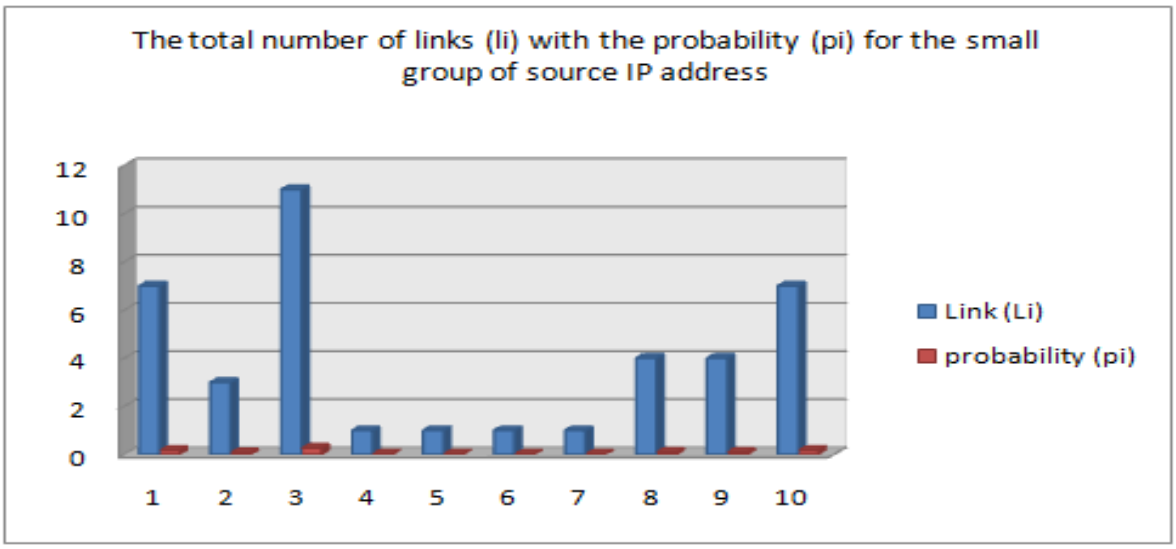
As per the bipartite graph definition, we consider the two vertices as Source IP address and Destination Port number. Both are disjoint sets and they are also independent sets. Fig shows the relationship of Source IP address to Destination Port numbers.



	1	2	3	4	5
Dst. Port No.	3868	5355	1900	1434	137
No. of edges	1	4	11	9	1
Frequency	0.024096	0.036145	0.222892	0.240964	0.433735

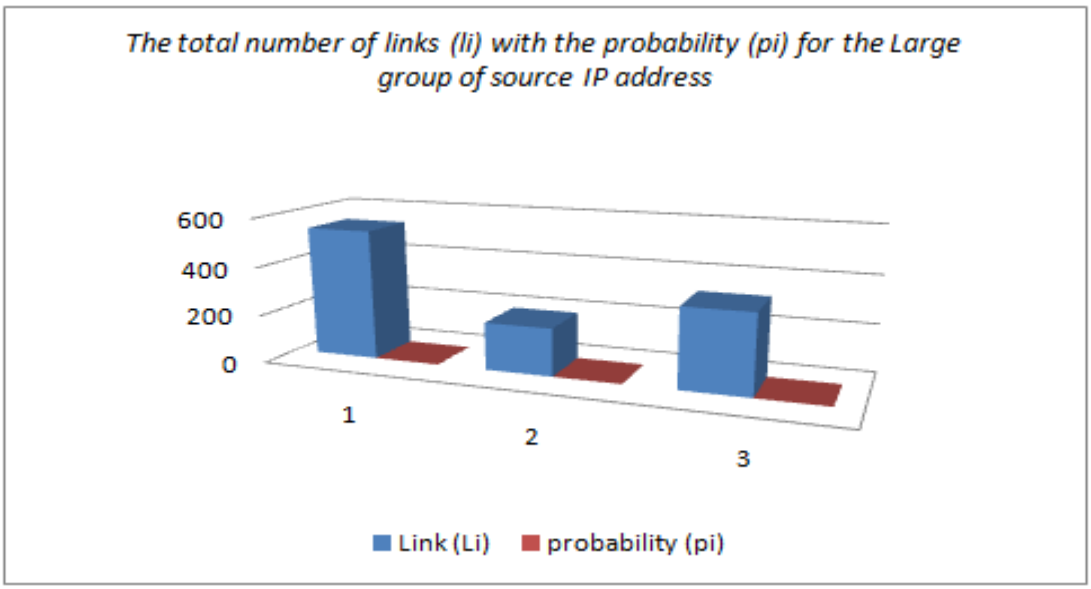
Result with respect to attack on the Destination Port.

Link	Li	7	3	11	1	1	1	1	4	4	7
probability	pi	0.175	0.075	0.275	0.025	0.025	0.025	0.025	0.1	0.1	0.175
		The total number of links (li) with the probability (pi) for the small group of source IP address									



Implementation of Information Entropy(IE) for the small group

Link	(Li)	538	197	328
probability	(pi)	0.506	0.185	0.308



Implementation of Information Entropy(IE) for the Large group

CONCLUSIONS AND FUTURE WORK

We have studied malware filter placement problems from an optimization perspective. After drawing the connection between traffic-weighted centrality measures and traffic measurements at routers, we chose a convex cost objective involving centrality measures. The first optimization problem we considered involves minimizing this cost subject to sampling and effective sampling rate constraints, as well as a constraint on the amount of traffic that can be filtered network-wide. Next we studied the case where instead of placing a hard upper bound on the quantity of filtering, we assign a cost to filtering and minimize a

sum of it and the cost metric derived earlier. We then minimized a different cost metric involving a sum of filter deployment and filtering costs less a utility measure under the same constraints. We found exact or approximate centralized and dynamic solutions to these optimization problems and simulated the resulting strategies. Network traffic data from the Abilene dataset was used in these simulations. We compared these strategies with benchmark approaches to network traffic filtering. The simulation results confirm that by applying optimization tools we can achieve lower costs in a variety of contexts and when traffic magnitudes change rapidly. There are several obvious extensions to this work. The optimization problems developed here should be solved in a decentralized manner for increased reliability and security. Various update algorithms could be considered when evaluating decentralized solutions. A natural extension to this paper would involve incorporating filter effectiveness with Bayesian analysis. This would al- low for a comparison between signature-based and anomaly- based filters. Constraints on the amount of signature-based and anomaly-based filtering could be set. Finally, we are planning to use the Abilene data toper- form more thorough simulations with the realistic Network Security Simulator (NeSSi).

Graph theory has been studied extensively in association with complex communication networks. We described basic concepts of graph theory and their relation to communication networks. Then we presented some optimization problems that are related to routing protocols and network monitoring and showed that many of the optimization problems are NP-Complete or NP-Hard. Finally, we explained some of the common tools used to generate network topologies based on graph theory.

REFERENCES

1) 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.
2) R. Deraison. Nessus scanner, 1999. Available at http://www.nessus.org.
3) D. Farmer and E. Spafford. The COPS security checker system. In USENIX Summer, pages 165–170, 1990.
4) S. Jajodia and S. Noel. Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response. In B. Bhattacharya, S. Sur-Kolay, S. Nandy, and A. Bagchi, editors, Algorithms, Architectures, and Information Systems Security. World Scientific Press, 2007.
5) S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, Managing Cyber Threats: Issues, Approaches and Challenges. Kluwer Academic Publisher, 2003.
6) P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. IEEE Security & Privacy Magazine, 4(6):85–89, 2006.
7) S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In CCS Workshop on Visualization and Data Mining for Computer Security04, 2004.
8) S. Noel and S. Jajodia. Understanding complex network attack graphs through clustered adjacency matrices. In Proceedings of the 21st Annual Computer Security Applications Conference, 2005.
9) S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency grpahs. In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03), 2003.

10) O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02), 2002.
11) L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In IFIP WG 11.3 Conference on Data and Application Security, 2008.
12) L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Computer Communications,29 (15):2917–2933, 2006.
13) Y. BEJERANO and R. RASTOGI, Robust Monitoring of link delays and faults in IP networks, In Proceedings of IEEE INFOCOM, (2003).
14) D. WADDINGTON, F. CHANG, R. VISWANATHAN, and B. YAO, Topology discovery for public IPv6 networks, ACM SIGCOMM Com- puter Communication Review, (2003).