Quantifying and Verifying Network Reachability

Amir R. Khakpour Alex X. Liu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
{khakpour, alexliu}@cse.msu.edu

ABSTRACT

Quantifying and verifying network reachability is important for security monitoring and auditing as well as many aspects of network management such as troubleshooting, maintenance, and design. Although attempts to model network reachability have been made, feasible solutions to computing network reachability have remained unknown. In this paper, we propose a suite of algorithms for quantifying reachability based on network configurations (mainly ACLs) as well as solutions for querying and verifying network reachability. We present a comprehensive network reachability model that considers connectionless and connection-oriented transport protocols, stateless and stateful routers/firewalls, static and dynamic NAT, PAT, etc. We implemented the algorithms in our network reachability analysis tool called Quarnet and conducted experiments on a university network. Experimental results show that the offline computation of reachability matrices takes a few hours and the online processing of a reachability query takes 0.075 seconds on average.

1. INTRODUCTION

For security and management reasons, the reachability among hosts that are physically interconnected is often limited by Access Control Lists (ACLs), *i.e.*, packet filters, configured on routers and firewalls. Correctly configuring ACLs is critical as it controls the reachability of the hosts in a network. However, ACLs are difficult to configure correctly. First, the rules in an ACL are logically entangled because of conflicts among rules and the resulting order sensitivity, and an ACL may consist of a large number (*e.g.*, thousands) of rules. Second, an ACL often consists of legacy rules written by different administrators, at different times, and for different reasons. Third, a path from one subnet to another often consists of many ACLs, which are often maintained by different administrators. Overall, correctly managing the ACLs in a large enterprise network is extremely difficult. It has been observed that the ACLs on the Internet often have errors [3]. An error in an ACL either creates security holes that will allow malicious traffic to sneak into a private network or blocks legitimate traffic and disrupts normal businesses, which in turn could lead to irreparable, if not tragic, consequences.

This paper concerns quantifying and verifying network reachability. Quantifying the reachability between any two subnets means to compute the set of packets that can traverse from one subnet to another based on network configurations. Querying and verifying reachability means to ask questions like "what can access what". They are useful in many aspects of network management such as network troubleshooting and maintenance, reachability monitoring, security monitoring and auditing, etc. Little work has been done on network reachability analysis and no concrete algorithm for computing network reachability has ever been proposed previously. Xie *et al.* presented a model of network reachability in their seminal work [3]; however, they give no algorithms for computing reachability (and of course no experimental results). Xie *et al.*'s network reachability model does not address dynamic NAT (Network Address Translation) and PAT (Port Address Translation), and does not take into account whether transport layer protocols are connectionless or connection-oriented. Furthermore, Xie *et al.*'s model is limited to describing the networks where each subnet connects to only one router because they model a network as a graph over only routers.

In this paper, we present Quarnet, a tool that comprises a suite of concrete algorithms for quantifying and querying network reachability. For quantification, Quarnet takes a network topology and the ACLs deployed on middleboxes as its input, and outputs reachability matrices that represent the lower-bound reachability (*i.e.*, the minimal set of packets that can traverse from a source to a destination at any time), instantaneous reachability (*i.e.*, the set of packets that can traverse from a source to a destination at a particular time), and upper-bound reachability (*i.e.*, the maximal set of packets that can traverse from a source to a destination at some time) for every pair of source and destination subnets. For querying, Quarnet allows network operators to ask questions about the reachability of the subnets in their network, *e.g.*, "which hosts in a subnet can access the mail server in another subnet?".

2. NETWORK MODELING AND REACHABILITY FORMULATION

In this paper, we model a network as a non-simple directed bipartite graph, where vertices represent subnets and middleboxes in the network. We use the term "subnet" to

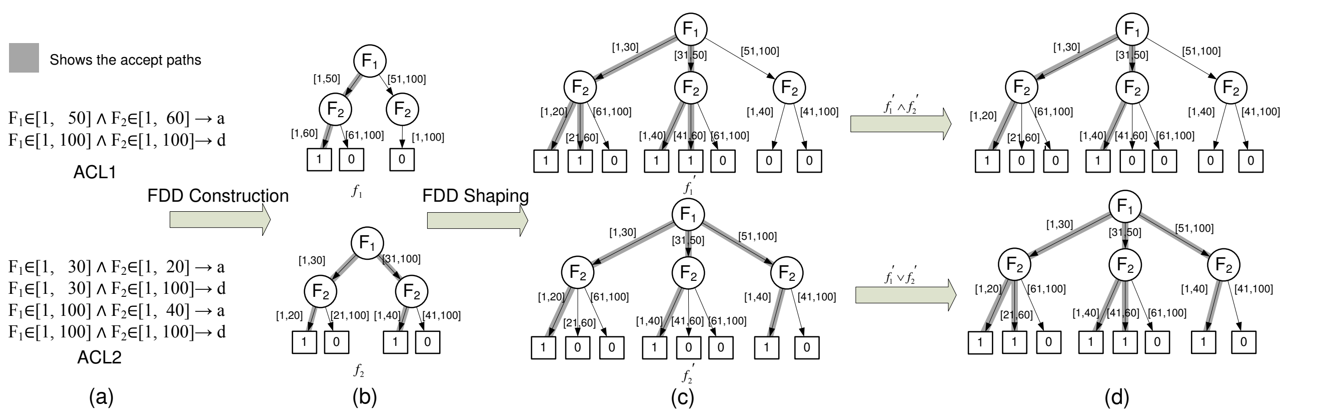
Shows the accept paths
F₁∈[1, 50] ∧ F₂∈[1, 60] → a
F₁∈[1, 100] ∧ F₂∈[1, 100] → d
ACL1
F₁∈[1, 30] ∧ F₂∈[1, 20] → a
F₁∈[1, 30] ∧ F₂∈[1, 100] → d
F₁∈[1, 100] ∧ F₂∈[1, 40] → a
F₁∈[1, 100] ∧ F₂∈[1, 100] → d
ACL2
FDD Construction
FDD Shaping
f₁ ∧ f₂
f₁ ∨ f₂
(a)
(b)
(c)
(d)

Figure 2: (a) Two ACLs (b) Two FDDs before shaping (c) Two FDDs after shaping (d) FDD logical AND/OR

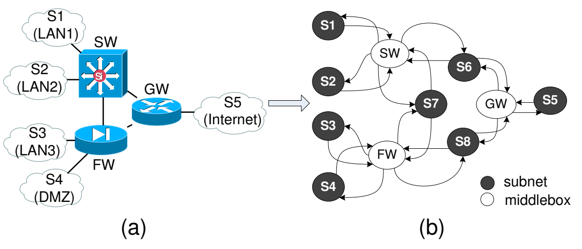

Figure 1: Example network topology

represent a set of adjacent subnetworks (*i.e.*, local area networks (LANs) or VLANs), where either they have the same reachability (*i.e.*, there is no ACL deployed between any two subnetworks in the set) or the reachability among the subnetworks is not a concern. The term "middlebox" refers to any networking device that can forward packets from one subnet to another, such as a network router, a firewall, a traffic shaper, or a L3 switch. Figure 1(a) shows a network with three middleboxes and four subnetworks. We then use this network model to formulate instantaneous reachability and lower-bound and upper-bound reachability, for connectionless protocols (considering one-way path reachability) and connection-oriented protocols (considering two-way path reachability) between each pair of subnets in the network. We extend our formulation for situations where NAT/PAT and stateful middleboxes are placed in the path between source and destination. We have considered different types of NAT, *i.e.*, static NAT and dynamic NAT. The formulation take advantage of the transformational function and its reverse function to calculate reachability.

3. ALGORITHMS FOR COMPUTING REACHABILITY

We represent network reachability as the six matrices shown below. We use n to denote the number of subnets, and z to denote the maximum number of paths between any pair of subnets.

1. $A^I_{CL}[1..n, 1..n, 1..z]$ where each element $A^I_{CL}[i, j, k]$ is the set of packets representing the instantaneous reachability from N_i to N_j on the k-th path for connectionless protocols.
2. $A^I_{CO}[1..n, 1..n, 1..z, 1..z]$ where each element $A^I_{CO}[i, j, k, k']$ is the set of packets representing the instantaneous reachability from N_i to N_j on the k-th data path and the k'-th signaling path for connection-oriented protocols.
3. $A^L_{CL}[1..n, 1..n]$ where each element $A^L_{CL}[i, j]$ is the set of packets representing the lower-bound reachability from N_i to N_j for connectionless protocols.
4. $A^L_{CO}[1..n, 1..n]$ where each element $A^L_{CO}[i, j]$ is the set of packets representing the lower-bound reachability from N_i to N_j for connection-oriented protocols.
5. $A^U_{CL}[1..n, 1..n]$ where each element $A^U_{CL}[i, j]$ is the set of packets representing the upper-bound reachability from N_i to N_j for connectionless protocols.
6. $A^U_{CO}[1..n, 1..n]$ where each element $A^U_{CO}[i, j]$ is the set of packets representing the upper-bound reachability from N_i to N_j for connection-oriented protocols.

To calculate all these matrices, we need to provide algorithms to efficiently calculate the set operations such as union and intersection of multi-dimensional packets. We use Figure 2 to briefly explain the core algorithms. In this scheme, we first use Firewall Decision Diagram (FDD) introduced in [1] as a data structure for representing the accepted and discarded packets of two example ACLs shown in Figure 2(a). The two example FDDs f_1 and f_2 are shown in Figure 2(b). We then reshape f_1 and f_2 to f'_1 and f'_2 such that they become identical except their decisions (Figure 2(c)). Finally, as shown in Figure 2(d) we do logical AND and OR operations for decisions of f'_1 and f'_2 to calculate the union and intersection of accepted packets, respectively. More details about how reachability matrices are calculated is available in [2].

4. ONLINE REACHABILITY QUERIES

After reachability matrices are calculated, we can use them as the engine for efficiently processing network reachability queries. We define an SQL-like language called *Structured Reachability Query Language* (SRQL) for specifying reachability queries. SRQL has the following format:

```
reachability_type $\mathcal{T}$
connection_type $\mathcal{O}$
select $\mathcal{F}$
where $(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d) \wedge (\mathcal{P} \in S_P)$
```

The reachability type $\mathcal{T}$ denotes the type of reachability, namely instantaneous (I), upper-bound (U), or lower-bound (L). The connection type $\mathcal{O}$ denotes the connection orientation of transport protocols, namely connection-oriented (CO) or connectionless (CL). When the reachability type $\mathcal{T}$ is upper-bound or lower-bound, the `select` clause $\mathcal{F}$ is a subset of packet fields $\{F_1, F_2, \cdots, F_d\}$; when $\mathcal{T}$ is instantaneous, $\mathcal{F}$ is a subset of fields $\{F_1, F_2, \cdots, F_d, \mathcal{P}\}$ where $\mathcal{P}$ denotes the attribute of "path". In the `where` clause, the predicate $(F_1 \in S_1) \wedge \cdots \wedge (F_d \in S_d)$ specifies the set of packets that this query is concerned with and $(\mathcal{P} \in S_P)$ specifies the set of paths that this query concerns. For example, SRQL query for the question "Through what paths the mail server in S4 on TCP port 25 is accessible from S1?" is the following:

```
type I
protocol CO
select P
where (S ∈ S1) ∧ (D ∈ MailServer) ∧ (DP ∈ 25) ∧ (PT ∈ TCP)
```

The answer to some questions may be the union or intersection of multiple SRQL query results. For example, the answer for the question "Which hosts in S1 can access the Mail Server in S4 on both UDP and TCP port 25 via any path from S1 to S4?" is the intersection of the results of the following two SRQL queries:

```
type L
protocol CO
select S
where (S ∈ S1) ∧ (D ∈ MailServer) ∧ (DP ∈ 25) ∧ (PT ∈ TCP)
```

```
type L
protocol CL
select S
where (S ∈ S1) ∧ (D ∈ MailServer) ∧ (DP ∈ 25) ∧ (PT ∈ UDP)
```

5. EXPERIMENTAL RESULTS

We implemented Quarnet in C++ and evaluated it on a university campus network. We focused the measurement on the execution time and memory usage of Quarnet. We concluded that Quarnet is sufficiently efficient to be used in practice as off-line computation, although computing network reachability is a resource consuming task in nature.

This campus network consists of 48 subnets interconnected in a hierarchical topology. Note that one subnet may contain multiple Virtual Local Area Networks (VLANs) and there are no ACLs among VLANs. In this network model, there are 49 subnets that are connected by 192 links through 2401 paths. Also, the total number of VLANs is 399 that are protected by 98 ACLs, where each ACL contains 573 rules in average (total number of rules: 56189). Among the 98 ACLs, 14 of them are original and the rest are generated based on the statistical features of the original ones and the subnet addresses.

	# of FDDs	Time (mins)	RAM (MB)
FDD construction	192	2.1	1276
One-hop path calculation	96	0.3	106
FDD matrix calculation	2352	11	1505
Reachability matrices calculation for connection-less protocols	2352	661	2400
Reachability matrices calculation for connection-oriented protocols	2352	1	800
Total execution	7344	675	4700

Table 1: Results on computing reachability matrices

As our reachability query engine uses FDDs as its core data structures, we evaluated the performance of online query processing by performing randomly generated queries over large FDDs with millions of nodes. Our experimental results in Table 2 show that our online reachability query engine is very efficient. For example, over an FDD with 2 million nodes, which is similar to the size of the FDDs uses in the online query engine built for the university campus network that we experimented, the average processing time for a query is 0.075 seconds, although some queries (less than 1%) take 2-3 seconds. The existence of some outliers is because some randomly generated queries may cause the engine to search through a large portion of the FDD.

FDD Size (# nodes)	Average Query Processing Time	Outliers
0.5 million	0.032s	1% of queries: 0.5s ∼ 1s
1 million	0.049s	0.8% of queries: 0.8s ∼ 1.5s
2 million	0.075s	1% of queries: 2s ∼ 3s

Table 2: Performance of online query processing

6. REFERENCES

[1] M. G. Gouda and A. X. Liu. Structured firewall design. *Computer Networks Journal (Elsevier)*, 51(4):1106–1120, March 2007.

[2] A. Khakpour and A. X. Liu. Quarnet: A tool for quantifying static network reachability. Technical Report MSU-CSE-09-2, Michigan State Univesity, Dept. of Computer Science and Engineering, January 2009.

[3] G. Xie, J. Zhan, D. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford. On static reachability analysis of IP networks. *Proc. IEEE INFOCOM*, 3:2170–2183, March 2005.