Using DFA to enforce an Access Control List

Joseph Schmidt

February 6, 2017

1 Overview

In computer science, Distinct Finite Automata or DFA are state machines used to validate whether a string of symbols is accepted by a language. This concept can be applied to many facets of computer science, including cyber security, in which it is possible to use a DFA to enforce an Access Control List or ACL on a networked device which restricts the network traffic on the device from potential threats. The goal of this project is to create DFA dynamically based on a given ACL in order to restrict traffic on a device based on the source IP addresses of IP packets.

2 Implemetation Plan

The author intends to use Python to program a system that will enforce an ACL by creating a DFA for an inputted ACL. First, the program will take an input file (`.txt`) which have commands similar to implementing an ACL in the CISCO networking operating system:

- The `permit` command followed by an IP address will allow the traffic from the source device to the host. This will also be usable with wild cards so multiple devices on a particular network can be allowed to communicate with the host.
- The `deny` command followed by an IP address will block traffic from the source device to the host. This will also be usable with wild cards so multiple devices on a particular network can be blocked.
- All traffic will be denied by default. As an example, if there are no commands in the inputted file, then all traffic will be denied.
- Essentially, this means there will be two types of ACLs: one which will permit certain addresses and then block all others, or one that will deny particular addresses while allowing all others. Additionally, a `permit any` command will permit all traffic and `deny any` will deny all traffic (the default).

Next, based on the input file, a DFA will be created to enforce the commands given. It will create DFA based on each `permit` and `deny` command, giving precedent to the first type of command given. For permitted addresses, an end state is achieved if the DFA validates the source IP address given by the command. Denied addresses will work similarly for `deny` commands; if the DFA validates the source IP address then the end state is reached.

Finally, a file with network traffic captured by Wireshark will be inputted into the system, and will read the source IP addresses of each packet in the file. The DFA will then permit or deny each packet based on it's source IP address, and the system will output the number of denied or permitted packets, as well as which packets were denied or permitted.

3 Conclusion

Overall this project will show how DFA can be used in cyber security, and how it can be used to protect devices on a network from possible threats. If there are any futher inqueries, please contact the author via email at Joseph.Schmidt2@marist.edu.