Using DFA to enforce an Access Control List

Joseph Schmidt

April 4, 2017

1 Abstract

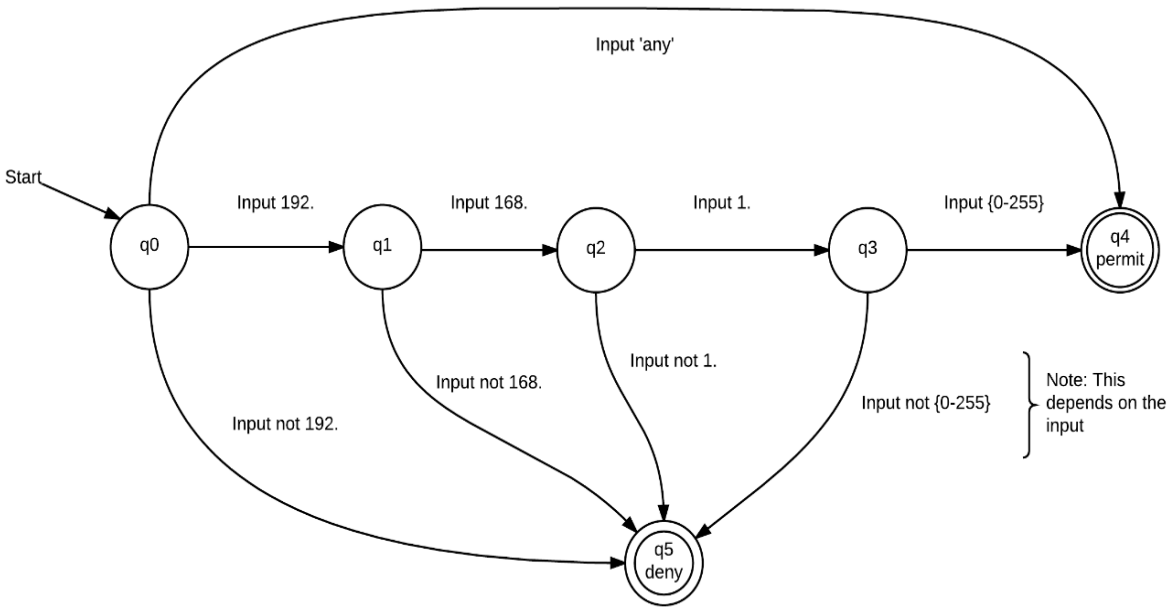
Deterministic finite automata can be used to validate data from a source. In cybersecurity, this can be used to validate and accept certain data sources on a network based on an access control list. A state machine can then be created from this access control list to enforce the validation of data as it comes to a device.

2 Introduction

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 (ACL) on a networked device which restricts network traffic from potential threats. This project creates DFA dynamically based on a given ACL in order to restrict traffic on a device based on the source IP addresses of IP packets from a WireShark capture file, within the local address scheme of 192.168.1.X, where X is any number between 0-255. The following will include the specific dectails of the project, how it is used, it's limits, and what can be done in the future with the project.

3 Detailed System Description

This project adheres to the DFA diagram below. This diagram is processing the input after either a `deny` or `permit` statement, taken from an ACL. This reflects how the DFA will be created as statements are processed from an ACL; the string following the `deny` or `permit` statement, which will either be a 192.168.1.X address (other address schemes currently not accepted) or the word `any`, is what the DFA will be matching on. For each of the `deny` or `permit` statements, there will simply be two DFA created based on a given ACL: one for the `deny` statements and one for the `permit` statements. However, since this project denys by default, the `deny` DFA will process commands after the `permit` DFA. This allows the permitted packets to pass through so there is no delay in the network communication.



The diagram only shows the `permit` DFA, though if state q4 was `deny` and q5 was `permit`, this diagram would be for the `deny` DFA. Again, this is reflected in processing ACLs, though creating the DFAs based on the valid ACL and the process of matching sources IP addresses from a WireShark capture file is still in development. Therefore the current inplementation will only validate an inputted ACL when the `main.py` file is ran. Lastly, as shown above, if the `permit any` statement is declared, any other states of the DFA will be overridden and any IP address will be permitted. Similar behavior should be expected when the `deny any` statement is declared.

4 Requirements

Any system wanting to use this project should have the following:

- Have at least 2GB of RAM and 25GB of memory on your system.
- Have Python 3.5.0 installed on the system.
- Ensure all required libraries found in the `requirements.txt` file of project are downloaded and installed (though this should be done automatically with the `Makefile` for the project).

5 Literature Survey

The author was inspired by the Cisco IOS software and its implementation of access control lists [1]. This work is similar in that it can take an ACL and enforce it, however this project does so with DFA using a graph data structure, as well as provide the user with information on what IP packets are being permitted and denied, such as the packet itself and the amount of permitted and denied packets.

6 User Manual

Please see `rules.txt` in the project for the rules this program follows regarding ACL processing and other details. Test cases for these rules, which are in the form of sample ACLs, can be found in the `/acl` folder of this project. The inputted ACL can be changed by modifying the `readacl.py` file. Further development will allow for accepting the ACL as an argument.

7 Conclusion

Overall this project shows how DFA can be used in cyber security, and how it can be used to protect devices on a network from possible threats. This idea can be further extended to work for all IP addresses, wildcards, and other features similar to Cisco IOS ACL functionality. Furthermore, this shows how effective state machines can be for validating data so it does not harm the device receiving a transmission. If there are any futher inqueries, please contact the author via email at Joseph.Schmidt2@marist.edu.

References

[1] Cisco IOS. Computer Software. Latest Version. Cisco. Accessed April 3, 2017. http://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-software-release-15-5-3-m/model.html.