Developing a Fake Classifier Node for DIFFUSE

Dzuy Pham*, Jason But
Centre for Advanced Internet Architectures, Technical Report 160422A
Swinburne University of Technology
Melbourne, Australia
dhpham@swin.edu.au, jbut@swin.edu.au

***Abstract*—DIFFUSE (Distributed Firewall and Flow-shaper Using Statistical Evidence) is a network prioritisation system which implements Machine-Learning (ML) based techniques to classify and subsequently prioritise flows. Inefficient testing methodologies impacts on future development, making it difficult to upgrade and expand. We have developed a Fake Classifier Node (FCN) that enables manual creation of DIFFUSE messages, bypassing the ML components, leading to more efficient testing and more reliable results. Tests confirm that the FCN is able create DIFFUSE messages and that they are correctly processed by existing DIFFUSE Action Nodes.**

I. Introduction

Machine Learning (ML) based classifiers have made significant progress recently and are steadily moving network management towards automation [1]. DIFFUSE (Distributed Firewall and Flow-shaper Using Statistical Evidence) [2], a network prioritisation scheme constructed around ML-based techniques was developed to segregate network traffic by 5-tuple inspection into classes and consequently deploy prioritisation. In its current state, DIFFUSE is constrained to FreeBSD [2] and OpenWRT [3] systems and thus problematic to deploy and test.

Additionally, testing and verifying each component inside DIFFUSE is complicated. DIFFUSE lacks testing programs, making it difficult to extend and develop for. DIFFUSE requires live traffic to be analysed by a Classifier Node before a classification can be made and prioritisation implemented. As such the testing of Action Nodes - which administer the prioritisation - is unreliable due to the dynamic nature of the ML-based classifier, and the reliance on live flow classification.

To tackle these problems, we developed a Fake Classifier Node (FCN) that is able to test any Action Node and verify its functionality. The FCN is a python based tool that is able to send customisable DIFFUSE messages. The FCN sends classification messages to the AN via the command line interface, effectively bypassing the statistical analysis of live flows in a traditional CN. The ability to create a customisable message results in more consistent outcomes in the AN without being dependant on ML classification.

This paper is organised as follows. Section II will provide more detail on DIFFUSE and its structure, as well as explaining some of the challenges in extending DIFFUSE. Section III outlines the design of the Fake Classifier Node and the structure of AN compatible messages. While section IV covers the verification of the FCN and testing methodology. Finally, section V concludes with some remarks and future development opportunities.

II. DIFFUSE

DIFFUSE [2] is a system which uses statistical analysis and Machine Learning based techniques to first classify, and then prioritise selected flows in the network. DIFFUSE is split into two main components, the Classifier Node (CN) and the Action Node (AN).

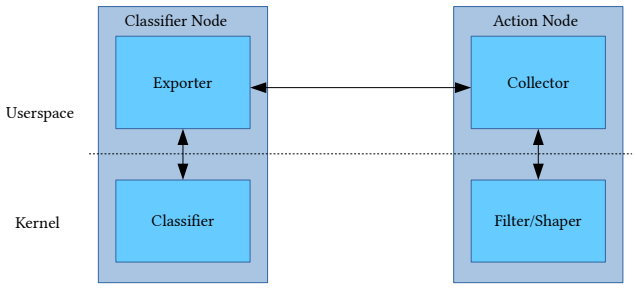


Figure 1. DIFFUSE Components

A. Classifier Node

The CN classifies flows using ML-based techniques based on statistical analysis of similar flows. The CN

*This work was performed while the author was an undergraduate summer intern under the supervision of Dr Jason But.

is made up of two components, 1) the classifier which performs statistical analysis to assign a particular flow to a class, and 2) the exporter which sends outbound messages to a specified AN.

The CN runs within the kernel using IPFW [4] to collect packets for processing. The CN filters packets into flows based on the 5-tuple to perform per-flow statistical analysis. The statistics are then passed to the ML-based classifier where they are matched to a class.

Once a classification is made, a Remote Action Protocol (RAP) (see Section III-A) packet is created to be sent by the exporter to a collector module in an AN to be acted upon. Communications with the AN can use a variety of transport layer protocols including TCP, UDP, and SCTP.

Currently CN implementations only exist for FreeBSD [2] and OpenWRT [3].

B. Action Node

The AN is also made up of two components, 1) the collector module which receives RAP messages from a CN, and 2) the filter/shaper which implements an action.

The action taken by the filter/shaper is based on the flow class as reported by the CN, and instructions based on the contents of an actions file that is loaded during the AN setup. The resultant action is then applied to Dummynet [5] in order to create the rule in the IPFW [4] table. All subsequent packets of this flow will now be (de)prioritised based on the classification.

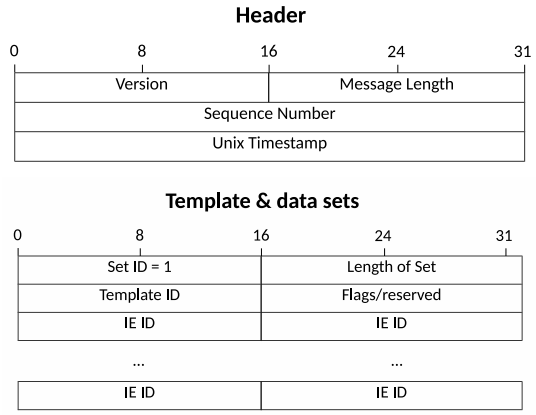
Current AN implementations also only exist for FreeBSD [2] and OpenWRT [3].

C. Extending DIFFUSE

One problem with further development of DIFFUSE is that testing the AN is complex. This is due to the nature of needing to:

1) Deploy a CN with a trained ML-based classifier
2) Generate live traffic flow, to create a classification
3) Wait for the CN to classify traffic and inform the AN to take action

To address this problem, we have developed the FCN to make future testing of any AN easier. The FCN is able to create RAP packets to classify flows based on command line parameters. This allows complete testing of all configured classes within an AN without the dependance on a full CN with ML-based classification.



*Information Element IDs can be found in [2]

Information Elements (IE)
Set ID = 256
Length of set
Export Name
Message Type
IPv4 Source Address
IPv4 Destination Address
Source Port
Destination Port
Protocol
Packet Count
Kilobyte Count
Class Tag (Contains: length, name, priority)
Timeout Type
Timeout
Action Name
Action Flag
Action Parameters

Information Elements in the order they appear in the RAP packet

Figure 2. Remote Actions Protocol layout

III. Design of the Fake Classifier Node

The Fake Classifier Node (FCN) generates packets that conform to the Remote Actions Protocol structure III-A. The AN can now be programmed directly bypassing the machine learning components completely. The FCN is coded in python and accessible from the command line. Flow classifications created by the FCN can be sent to any ANs that are listening.

The FCN is designed to be compatible with any AN, including the FreeBSD and OpenWRT. Although DIFFUSE allows a CN to send specific actions to the AN, this wasn't implemented into the FCN as that function

can also be performed by the actions file. The FCN can be used to either add or delete a flow classification from the AN. These classifications and any associated parameters can be specified from the Command Line Interface (CLI) (See Section III-B) and sent using the specified protocol to any AN.

A. Remote Action Protocol

The Remote Action Protocol (RAP) is an application layer protocol, the payload is formatted using the RAP protocol definition [2]. For typical UDP messages, the template is sent along with the data to create a flow, allowing the AN to know what the format of the data is before parsing its payload. For a TCP message, the template may be sent at the beginning of the connection and subsequent messages will only contain the flow classification information.

There exist three main components to the payload. For the FCN, we used the format as shown in Figure 2.

While DIFFUSE is capable of sending multiple flow classifications within a packet, the FCN setabds only one flow classification per instantiation. This simplifies the CLI and aids in the debugging of the AN, as only one action will occur per message from the FCN.

B. About the Program

The FCN is written in Python and connects to any AN using either UDP or TCP. A CLI was implemented to allow customisable RAP messages. Once the parameters have been set, the FCN will create the RAP message, connect to a specified AN and send the message to the AN.

The FCN command line options are summarised in Figure 3). A more detail description can be found in Table I.

IV. Verification

Verifying that the FCN is compatible with a DIFFUSE AN confirms its functionality and that it is a viable tool for future development of DIFFUSE. Each function of the FCN was made sure to work correctly with the original DIFFUSE with multiple different parameters set.

A. Testing the FCN

The FCN needs to be able to correctly perform the output function of a DIFFUSE CN. In order to test this, we need to confirm that messages from the FCN are being properly implemented in a DIFFUSE AN. This is done by sending flow update messages for nominated 5-tuples to an AN, and then checking to see if the appropriate rules are being created in IPFW.

To test the FCN, a copy of FreeBSD 9.0 was loaded onto a virtual machine, with DIFFUSE and Dummynet [5] installed. The IPFW table was initialised using the commands in Figure 4. In this example, flows allocated to queue 2 are prioritised over flows allocated to queue 1, the default rule allocates all flows to queue 1. The AN was configured to listen for RAP messages on UDP port 5000. The AN configuration file (see Figure 5) specifies two traffic classes, **myclass** and **other**.

```
kldload dummynet
ipfw pipe 1 config bw 100Mbit/s
ipfw queue 1 config pipe 1 weight 100
ipfw queue 2 config pipe 1 weight 10
ipfw add 900 accept ip from any to any
in via em0
ipfw add 2100 queue 1 ip fron any to
any via em0
```

Figure 4. IPFW Initial Configuration

```
default queue 2
myclass:1 queue 1
other:1 queue 2
```

Figure 5. DIFFUSE Action File

We then used the FCN running on a second computer to send different messages to the AN. We incremented the RAP sequence number for each message. In total, five different messages were sent using the commands in Figure 6. Figure 7 shows the contents of the IPFW table after the FCN commands were executed. We can visually verify that the appropriate rules have been added to the IPFW table.

We also tested sending a flow removal command via the FCN and verifying that the corresponding rule was deleted from the IPFW table.

Our testing indicates that the RAP messages formed by the FCN are properly constructed. All messages resulted in rules being properly processed by the AN and subsequent prioritisation rules being installed/removed. This gives us confidence that the FCN can be used in future testing of any DIFFUSE compatible ANs.

V. Conclusions

This report focuses on addressing some of the challenges in testing DIFFUSE. In order to facilitate future development of DIFFUSE, we created a Fake Classifier Node to easily and efficiently test Action Nodes. The FCN exists to minimise unexpected variables that can

Table I
FCN PARAMETERS

Command	Name	Description
–i	Source IPv4	Set the Source IPv4 address for matching. (5-tuple)
–j	Destination IPv4	Set the Destination IPv4 address for matching.(5-tuple)
–u	Protocol Type	Set the protocol type number for matching. The full list of protocols and their corresponding numbers can be found here [6]. (5-tuple)
–k	Source Port	Set the Source Port number for matching. (5-tuple)
–l	Destination Port	Set the Destination Port number for matching. (5-tuple)
–s	Sequence No.	The sequence number determines the order RAP packets were sent to the AN. The FCN requires the sequence number increased with every message sent for DIFFUSE or the message will be ignored.
–e	Export Name	The name of the exporter, which sent the packet in the CN.
–c	Class Name	The Classification Name given for this set of 5-tuple
–a	Msg Type	0 - AN adds the single rule to the rule table 1 - AN deletes all rules which match the 5-tuple 2 - AN deletes all rules in the flow table (Only available in RAN)
–t	Timeout Value	Time the flow rules in the AN will expire (Default: 60 seconds)
–n	Priority Match	The order in which flow table rules would match, higher numbers will match first and the action coresponding to the rule applied
–x	Host IPv4	IPv4 Address of the AN we want to connect to
–y	Host Port No.	Port Number of the AN we want to connect to
–z	Host Protocol	Protocol used to connect to the AN

```
usage: fakecnode.py [-h] [-i Source IP] [-j Destination IP] [-s Sequence Number] [-k Source
port] [-l Destination port] [-u Protocol Type] [-a Msg Type] [-t Timeout Value] [-e Export] [-c
Class] [-n Priority] [-x Host] [-y Port] [-z Proto]
```

Figure 3. FCN CLI

occur from a ML-based CN by performing the same functions as the CN but in a controlled environment. Testing of ANs no longer requires a working CN as they are replaced by the FCN.

Our testing using existing DIFFUSE Action Nodes demonstrates that the FCN is capable of creating customisable AN compatible messages. These messages were successfully processed by a working AN and prioritisation was able to be implemented by the AN.

Our work will allow testing of any ANs irrespective of their operating system. This also aids in the development of ANs in environments that currently do not have a working CN. Testing of multiple ANs can be made more efficient, whereby one FCN is able to easily program multiple installed ANs.

REFERENCES

[1] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4738466

[2] S. Zander and G. Armitage, "Design of DIFFUSE v0.4 - Distributed Firewall and Flow-shaper Using Statistical Evidence," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 110704A, 04 July 2011. [Online]. Available: http://caia.swin.edu.au/reports/110704A/CAIA-TR-110704A.pdf

[3] N. Williams and S. Zander, "Real Time Traffic Classification and Prioritisation on a Home Router using DIFFUSE," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 120412A, 12 April 2012. [Online]. Available: http://caia.swin.edu.au/reports/120412A/CAIA-TR-120412A.pdf

[4] U. J. Antsilevich, P.-H. Kamp, A. Nash, A. Cobbs, and L. Rizzo, "ipfw – user interface for firewall, traffic shaper, packet scheduler, in-kernel nat," *FreeBSD System Manager's Manual*, 2011. [Online]. Available: https://www.freebsd.org/cgi/man.cgi?query=ipfw&manpath=FreeBSD+9.0-RELEASE

[5] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1764876

[6] IANA, *Protocol Numbers*, Std., 2016. [Online]. Available: http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

FCN CLI INPUTS

```
python fake_cnode.py -i 10.0.0.1 -j 10.0.0.1 -k 8000 -l 80 -t 60 -s 10 -c myclass -x
192.168.1.2 -y 5000 -z udp
python fake_cnode.py -i 10.0.0.2 -j 10.0.0.1 -k 22 -l 22 -t 60 -s 11 -c other -x 192.168.1.2 -y
5000 -z udp
python fake_cnode.py -i 10.0.0.3 -j 10.0.0.1 -k 22 -l 22 -t 60 -u 17 -c myclass -s 12 -x
192.168.1.2 -y 5000 -z udp
python fake_cnode.py -i 10.0.0.4 -j 10.0.0.1 -k 1234 -l 1234 -t 60 -u 1 -c unknown -s 13 -x
192.168.1.2 -y 5000 -z udp
```

Figure 6. FCN Verification Diffuse

Output from DIFFUSE at 192.168.1.2

```
usage: ipfw show
00900    0    0 allow ip from any to any in via em0
01000    0    0 queue 1 tcp from 10.0.0.1 8000 to 10.0.0.1 dst-port 80 keep-state
01001    0    0 queue 2 tcp from 10.0.0.2 22 to 10.0.0.1 dst-port 22 keep-state
01002    0    0 queue 1 udp from 10.0.0.3 22 to 10.0.0.1 dst-port 22 keep-state
01003    0    0 queue 2 icmp from 10.0.0.4 1234 to 10.0.0.1 dst-port 1234 keep-state
02100    0    0 queue 1 ip from any to any via em0
```

*Bold Text shows that these rules were added after CLI inputs

Figure 7. DIFFUSE IPFW OUTPUT