Heriberto Camacho Torres

hcamacho94@gmail.com

Prof. Humberto Ortiz

humberto.ortiz@upr.edu

Computer Science Department - Rio Piedras

Universidad de Puerto Rico, Recinto de Rio Piedras

May, 2019

Learning Anomaly Detection with SiLK and Python

ABSTRACT

The detection of anomalies in and entire network is an incredibly daunting task, even with today's advancements in technologies. To that end, this research experience was aimed at learning techniques used for scanning and processing Network Flow data. By use of SiLK and Python, information was gathered that revealed how strong a presence each source IP address has in the network. This results can be of use as the foundation for the use of the Subspace method, which uses its a metric for determining which flows are anomalous, and which aren't.

INTRODUCTION

Network Anomalies

Anomalies are those elements that behave differently than how they normally would on normal conditions. In some cases, these occur as freak accidents, or as a result from randomness, and are deemed outliers. In a network though, anomalies can serve as identifiers of different events occurring at a given time.

Anomaly pattern detection

Since most cyber attacks can't be caught in real time. Our current technology allows us to read flow data and find ways to makes suspicious behaviour stand out.

General Anomaly detection problem

Presently, the biggest challenge is processing said flow data in real time. This is a robust data set, that can yield many results depending on how the information is filtered. This is the current task to be done, employing techniques that can be used throughout the data, and get specific results. To that end we explored the use of the subspace method.

METHOD

Use of SiLK and Python

On this project, the System for internet-Level Knowledge (SiLK), was used. This is a framework for processing flow data, developed by CERT Situational Awareness Group. It allows to extract desired flow data to be processed, by it command pipeline and/or python code. The support Python provides, allows a greater degree of freedom in how the information is extracted. It can then be used on the extracted data.

SiLK argument pipeline

The following command arguments allow the flows to be displayed on the terminal.

```
heriberto@hulk:/data/heriberto$ rwfilter --start-date=2015/02/26:13 --end-date=2015/02/26:16 --senso
r=S0,S1 --type=all --print-volume --threads=4 --pass-destination=stdout --site-config-file=/data/con
f-v9/silk.conf --protocol=0- | rwcut --fields=1,2,3,4,5 | head
```

Exporting and processing netflow data

This command line argument allows to export the flow data in a text format, to then be processed. This is a necessity since flow data is saved in binary format.

```
heriberto@hulk:/data/heriberto$ rwfilter --start-date=2015/02/26:13 --end-date=2015/02/26:16 --senso
r=S0,S1 --type=all --print-volume --threads=4 --pass-destination=stdout --site-config-file=/data/con
f-v9/silk.conf --protocol=0- | rwsort --fields=sIP | rwcut --fields=sIP,packets > sIPflows.out | les
s
```

As a starting point to begin the information processing of the data flows, a program was made that would read them, and store the relevant information in a dictionary. In this particular case, source IPs were the ones being recorded, and tallied.

```
import numpy as np
import re
import string

#This program reads the flow data, extracting and saving all Source IPs in a dictionary
#The goal is to record the number of occurrances of each sIP present.
def main():

    with open('sIPflows.out', 'r') as record:

        #Dictionary for storing the data source IP data
        #It is then read and the "|" simbols are removed
        sIP = {}
        o = record.read()
        test = re.split(r"\|", o)

        #For loop that cycles through the sIPs, strips the backspaces present
        #and keeps a tally of each one present in the dictionary
        for b in range(len(test)):

            if b%2 == 0:
                if string.lstrip(test[b]) not in sIP:
                    sIP[string.lstrip(test[b])] = 0
            else:
                sIP[string.lstrip(test[b-1])] += int(test[b])

        #Prints dictionary information stored
        for x, y in sIP.items():
            print(x,y)

if __name__ == '__main__':
    main()
```

RESULTS

Show processed flow data

The information provided by our flows are has two IPs stand out from the rest, as having the most presence in the network. This serves as a starting point of where to look for anomalies, or lack there off. Investigating the activities where these are involved can shed light into possible anomalous behavior.

```
('', 0)
('136.145.231.19', 12211)
('10.255.87.18', 3)
('136.145.231.13', 17062)
('136.145.231.12', 1871586)
('136.145.231.11', 5522)
('136.145.231.10', 2671)
('136.145.231.17', 5269)
('136.145.231.16', 5294)
('136.145.231.15', 12009)
('136.145.231.31', 3477)
('136.145.231.33', 20720)
('136.145.231.18', 6444)
('136.145.231.35', 44215)
('136.145.231.34', 8669)
('136.145.231.37', 81692)
('136.145.231.36', 18619)
('192.168.88.1', 198)
('136.145.231.22', 5493)
('136.145.231.20', 2073)
('136.145.231.21', 12923646)
('0.0.0.0', 5)
('136.145.231.40', 2827)
('136.145.231.41', 165679)
('169.254.188.44', 8)
('169.254.241.158', 11)
```

Use of SiLK

SiLK, is a powerful tool for managing the great amount of information that is present in a flow. The program showed only uses a small part of the information gathered, where only the source IPs were used, in the 3230 records stored. Albeit small, it can possibly yield a lot of information even more so with other functionalities and uses being explored in the future.

sIP	dIP	sPort	dPort	pro
136.145.231.34	224.0.0.252	0	22	2
136.145.231.10	224.0.0.251	0	22	2
136.145.231.12	10.255.231.2	0	2048	1
136.145.231.12	10.255.231.12	0	2048	1
136.145.231.34	239.255.255.250	0	22	2
136.145.231.12	10.255.231.11	0	2048	1
136.145.231.21	224.0.0.251	0	22	2
136.145.231.12	10.255.231.4	0	2048	1
136.145.231.41	5.9.8.132	22	30081	6

	Recs	Packets	Bytes	Files
Total	3230	100384	67243223	11
Pass	3230	100384	67243223	
Fail	0	0	0	

CONCLUSION

The program showed which IPs had the most presence in the network. Two of these take a huge portion of the network during the period of time being explored. Further study can reveal why that is the case, and can serve as a base for pointing anomalies in the network.

FUTURE WORK

Future work will explore more uses of the SiLK framework, with the goal of implementing the subspace method for detecting anomalies. This aims to address the bigger problem of network wide scanning for anomalies as fast as possible.

REFERENCES

[1]Lakhina, Anukool, Mark Crovella, and Christiphe Diot. "Characterization of network-wide anomalies in traffic flows." *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004.

[2]Krystosek, P; Ott, N. M. ; Sanders, G ; Shimeall, T. (2018). *Network Traffic Analysis with SiLK*. Pennsylvania USA: Carnegie Mellon University.

[3]Colón-Rosado, Bianca; Ortiz-Zuazaga, Humberto (2015): Techniques for Anomaly Detection in Network Flows. figshare. Journal contribution.