HAL
archives-ouvertes.fr

Mining NetFlow Records for Critical Network Activities

Shaonan Wang, Radu State, Mohamed Ourdane, Thomas Engel

▶ **To cite this version:**

Shaonan Wang, Radu State, Mohamed Ourdane, Thomas Engel. Mining NetFlow Records for Critical Network Activities. 4th International Conference on Autonomous Infrastructure, Management and Security (AIMS), Jun 2010, Zurich, Switzerland. pp.135-146, 10.1007/978-3-642-13986-4_20 . hal-01056635

HAL Id: hal-01056635
https://hal.inria.fr/hal-01056635

Submitted on 20 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution| 4.0 International License

Mining NetFlow Records for Critical Network Activities

Shaonan Wang[1], Radu State[1], Mohamed Ourdane[2], and Thomas Engel[1]

[1] Faculty of Science, Technology and Communication, University of Luxembourg
{shaonan.wang,radu.state,thomas.engel}@uni.lu
[2] P&TLuxembourg
{mohamed.ourdane}@dt.ept.lu

Abstract. Current monitoring of IP flow records is challenged by the required analysis of large volume of flow records. Finding essential information is equivalent to searching for a needle in a haystack. This analysis can reach from simple counting of basic flow level statistics to complex data mining techniques. Some key target objectives are for instance the identification of malicious traffic as well as tracking the cause of observed flow related events. This paper investigates the usage of link analysis based methods for ranking IP flow records. We leverage the well known HITS algorithm in the context of flow level dependency graphs. We assume a simple dependency model that can be build in the context of large scale IP flow record data. We apply our approach on several datasets, ranging from ISP captured flow records up to forensic packet captures from a real world intrusion.

1 Introduction

The monitoring of large network traffic volumes is limited by the existing technological solutions. Monitoring high speed 40 Gbps links is challenged by the already existing work charge on the routing data plane. One of the few activities that can be done is limited to recording and analyzing flow records. IP flow records are simple information records capturing the source, destination, the associated ports, the traffic volume and additional time stamps and flow related status. The natural question is how should these pieces of information be processed. On one side, the number of flow records is huge even for small sized edge routers, and on the other side it's not obvious what information should be analyzed. We have considered this research question in this paper. The main contribution of our paper is twofold: we propose a simple dependency model for IP flow records and show how link based analysis can reveal interesting flow events. We will use in this paper the words IP flow records and NetFlow records interchangeably. We have validated our approach using the proprietary NetFlow data format, but our method is general and can be applied to any flow record format. We aimed in this paper at identifying relevant flow records, where by relevant we understand the records that have generated ulterior network activity. We don't consider that a flow matching a specific signature (application level or

based on the involved IP addresses) is relevant per se, but we do consider that flows, having triggered an important follow-up network activity, are relevant. The notion of triggering is linked to a potential dependency relationship among flow records. The best illustration for this is the case of an attacker breaking in over an SSH account. While the SSH related flow traffic is in general not relevant, in this case this could be the case if follow-up activities of the compromised host will be observed: large scale network scanning, rootkit downloading, massive SMTP traffic or botnet membership. For scoring such relevant IP flow records and understanding the most active activities on the network, our approach consists of two major steps. Firstly, with a simple yet efficient dependency model, we discover the causality dependency between NetFlow records. Then, to facilitate analyzing the overwhelming scale of NetFlow dependency graph, we automatically select the most relevant NetFlow records using the link analysis algorithm HITS [12]. To the best of our knowledge, this is the first attempt to apply HITS algorithm from the web search and bibliometrics domain, in the field of network monitoring. The experiment result shows that HITS algorithm suits well the task of ranking most relevant flows from the perspective of network anomaly detection, bandwidth usage, etc.

The remainder of the paper is organized as follows. Section 2 provides an overview of our NetFlow ranking architecture. Section 3 presents the background of NetFlow and NetFlow collecting approaches. We explain and discuss our NetFlow dependency discovery engine in section 4, and in section 5, we rank flow records using HITS algorithm and interpret the results. We validate our flow ranking technique with various data sets in section 6. Section 7 summarizes the related work and section 8 concludes the paper and talks about future work.

2 High-level Architecture of FlowRank

Before showing the design and implementation details of each component, we provide a high-level architecture of our NetFlow ranking system. As illustrated in Figure 1, our ranking architecture consists of three components: NetFlow collector, dependency discovery engine, and rank engine. *NetFlow collector* collects NetFlow records either through border routers or dedicated probes. The *Dependency discovery engine* discovers dependencies among NetFlow records. The intuition of the dependency discovery is that if NetFlow A triggers NetFlow B, then destination address of NetFlow A matches source address of NetFlow B. NetFlow dependencies provide a global view of causalities among network traffic, thus is a valuable tool to detect the root cause of abnormal network traffics. To select the most dependent NetFlow records among the huge amount of dependencies discovered previously, *rank engine* ranks the relative importance of each NetFlow record using link analysis algorithm HITS [12]. Experiment results in the section 6 show that HITS is indeed appropriate for this task.

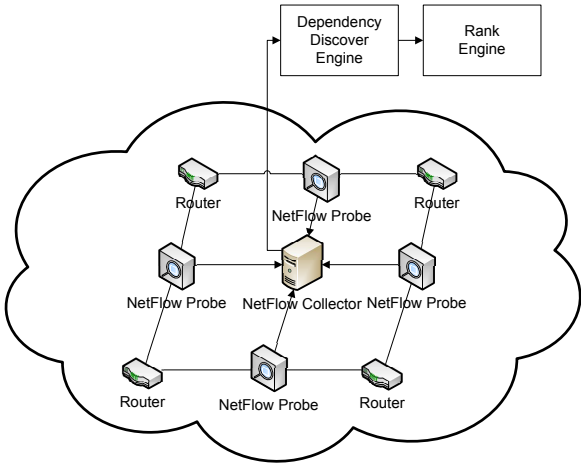


Fig. 1: Flow Rank Architecture

3 NetFlow Collection

3.1 What is NetFlow

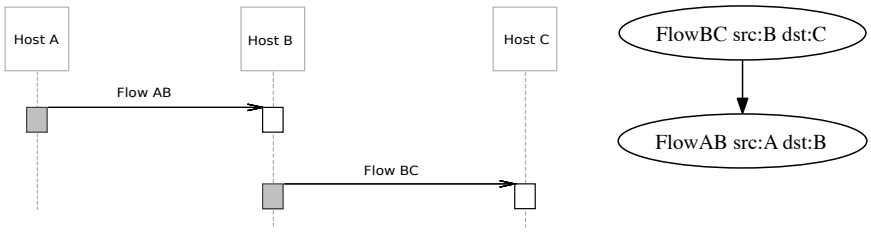
NetFlow is a proprietary protocol for collecting IP flow packets information on networks. Though some variant versions exist, a NetFlow record summarizes a network traffic flow as its source and destination IP addresses, source and destination ports, transportation protocol as well as the traffic volume transmitted during this flow session. NetFlow operates by creating new flow cache entries when a packet is received that belongs to a new flow. Each flow cache keeps track of the number of bytes and packets of similar traffic during certain period of time until the cache expires, then this information is exported to a collector. NetFlow provides a powerful tool to keep track of what kind of traffic is going on on the network, and are widely used for network monitoring. Most vendors support different flavors of similar flow monitoring approaches and a common standardization is done within the IETF IPFIX working group [6].

3.2 NetFlow Collection

One can collect NetFlow records either directly through border routers or using stand alone probes such as taps. Router based approaches require no extra hardware installation, but suffer from inaccurate report and degraded routing performance in case of traffic peaks. On the other hand, stand alone probe approaches provide reliable NetFlow record reports even in case of large volume of traffic, but require additional hardware installation in every link that needs to be observed and cause extra cost for maintenance.

4 Dependency Discovery

Causalities among netflow records contain valuable information for detecting the root cause of attacks as well as the most critical services that other services require to function properly. We assume that a NetFlow record depends on the other if the former is triggered by the latter. In our current model, we consider NetFlow record A causes NetFlow records B if after observing flow A arrives at a host H, within a predefined time window, we observe also flow B going out of the same host H. Figure 2 illustrates an example of dependencies between two flows. Figure 2a shows a message sequence chart in which there are three Hosts A, B, and C. After observing $FlowAB$ coming from $HostA$ to $HostB$, we observe $FlowBC$ coming from $HostB$ to $HostC$.



(a) NetFlow sequence chart (b) NetFlow dependency graph

Fig. 2: Flow dependency example

Two nodes in the dependency graph as shown in Figure 2b represent two NetFlow records. Each node is labeled with NetFlow record ID (labeled as rcdID), source IP and source port (labeled as src) and destination IP and destination port (labeled after dst). Record IDs suggest the time order in which the NetFlow records are observed. Small ID is observed earlier. That is, NetFlow 1 is observed before NetFlow 2. In addition, the destination IP address of NetFlow 1 matches the source IP address of NetFlow 2, according to our dependency model, we assume flow 1 triggers flow 2, in other words, flow 2 depends on flow1. We use a directed edge pointing from flow 2 to flow 1 to denote a dependency of flow 2 on flow 1.

After aggregating all the NetFlow causalities within a time window, our dependency model is able to discover causalities of a flow which depends on the arrival of many other incoming flows. The same holds for discovering all the successive NetFlow records of a NetFlow record which triggers many outgoing NetFlow records. This model fits our objective in terms that it reveals the in-depth causal connections among NetFlows on a network level.

While the simplicity and efficiency of this model suits well for analyzing network traffic online, the accuracy of discovered dependencies can be improved using more advanced techniques. Previous studies such as Barham [2] and Reynolds [15] have applied probability approaches to improve the accuracy of dependencies discovery, but computation overheads made online approach infeasible. Our experiments show that our model can achieve adequately acceptable results in identifying the most critical NetFlow records

5 Relative Importance Rank Engine

Rank Engine ranks the relative importance of NetFlow records based on the dependencies discovered by *dependency discover engine* using link analysis algorithm HITS [12].

5.1 HITS algorithm

Given a dependency graph, HITS ranks the relative importance of each node with two values: authority value and hub values. Authority value indicates the importance of a node in terms of how many other nodes are dependent on it. Hub value indicates how many important nodes a node points at, that is how many other nodes it depends on. Authority value and hub value are computed in a recursive manner. More precisely, given an n*n adjacency matrix A generated from a dependency graph of n nodes, where entry (i, j) is 1 if node i depends on node j, and 0 otherwise. With an all one vector as the initial value, HITS computes authority value a_i and hub value h_i for node i using the following equations iteratively.

$$a_i^{(t+1)} = \sum_{j:j->i} h_j^{(t)} \tag{1}$$

$$h_i^{(t+1)} = \sum_{j:i->j} a_j^{(t+1)} \tag{2}$$

where $j- > i$ indicates node j depends on node i. In other words, in round $(t + 1)$, authority value of node i is the sum of round t hub values of the nodes that pointing to node i and hub value of node i is the sum of round $t+1$ authority values of the nodes that node i points at. Rewriting the above equations in the form of authority vector av and hub vector hv, we can get:

$$av^{(t+1)} = A^T hv^{(t)} = (A^T A)av^{(t)} \tag{3}$$

$$hv^{(t+1)} = Aav^{(t+1)} = (AA^T)hv^{(t)} \tag{4}$$

With normalization to unit length at the end of each iteration, authority and hub vectors converge at the principle eigenvector for matrix $A^T A$ and AA^T [13].

5.2 Rank NetFlow records with HITS

Fed with the adjacency matrix of NetFlow dependency graph, HITS algorithm ranks each NetFlow record with an authority value and hub value. According to the ranks, we can classify NetFlow records into four categories: low authority and low hub value records, high authority and low hub value records, low authority and high hub value records, and high authority and high hub value records.

A low authority and a low hub value indicates that the NetFlow record is not well connected with other flows in the dependency graph, thus of little significance in terms of causal evidence. This does not necessarily mean that such a record is benign. It might be highly nefarious, but at least it does not trigger successive network activities.

A high authority value and low hub value characterize a flow that many other flows are dependent on, but which on its turn have no many dependencies on other flows. Such flows are prime candidates for representing the root cause and primordial events in case of malicious or suspicious usage.

A low authority value and a high hub value for a flow record will indicate a flow that did not trigger important follow up network activity, but has many dependencies on other flows. Such flow records might indicate either a flow that ended a set of activities or a potential false positive.

Finally, we might have flow records having both high authority value and a high hub value. The associated flows correspond to network traffic having had both important follow up network activities and depending on many previous flows. These flows should in principle correspond to important traffic.

6 Experiment and Evaluation

The key experiments that we did perform address the following questions:

- Can we use our method as a forensic tool? That is, given a packet capture for which we know the real evidence (using a manual annotation) can we reveal the important flows using our method?
- How many flows are ranked in the four categories (low and high authority value, low and high hub value) for real network traffic?

We have validated our approach on several scenarios, among which we will highlight one whose network traffic are publicly available [14]. These traces were captured on compromised honeypot, deployed by the honeypot project [3].

6.1 Attack description

The network schema of this attack scenario is shown in Figure 3. The compromised machine is located on a local area network, with IP addresses ranging from 172.16.1.1 to 172.16.1.250. An attacker (IP address 211.180.209.190) performs a simple operating system fingerprinting in order to learn the operating

[3] www.honeynet.org

system of 172.16.1.103. This is is done using a telnet: no user name, nor passwords are provided, the attacker aims just at getting the welcome banner of the remote host. Next, the attacker scans several machines for the portmap service. Several hosts run this service, among which the attacker can find the honeypot (172.16.1.108). These reply to the attacker. The attacker requests the port number of the RPC service stat. Her main objective is to launch an exploit against a vulnerable version of the [7] service. The honeypot is vulnerable to this exploit [4] and the attacker is able to compromise it. The result consists in a remote shell being opened on the TCP port 39168. Using this shell, the attacker can interact with the compromised machine. She will download a rootkit using the file transfer protocol. The server is 193.231.236.41 and is located in Romania. Additionally, the attacker sends two emails. These are the connections going to the SMTP servers (216.136.129.14 and 209.61.188.3). To summarize this attack: An attacker uses at least two different step stones to scan and attack the network. Once the attack is successful, the attacker will download additional malware (from another location repository) in order to maintain her privileges and access rights.

6.2 Experimental result

We extracted 48 NetFlow records from the packet capture. Table 2 summarizes the associated NetFlow records. Figure 4 illustrates dependencies among the NetFlows. Each node stands for a NetFlow record labeled with the associated NetFlow ID. Due to the page space limitation, we labeled each node with a NetFlow record ID number instead of the detailed NetFlow record tuples. Details of selected high ranking NetFlow records can be found in Table 1. It also lists the ranking results after running HITS based on the dependencies graph. The cumulative histogram shown in Figure 5a displays the ranking score distributions of the NetFlow records collected from honey pot attack scenario.

6.3 Rank distributions on backbone traffic

We have looked at the statistical distribution for both authority value and hub value over several datasets obtained from a large European Internet Service Provider. The table 1 summarizes the properties of the underlying flows. We did not consider larger datasets; because we aimed at comparing the distribution with respect to the honeypot scenario. Figures 5 and 6 address this comparison. Note that in order to expose the most important flows, we scale the authority and hub values of each dataset up so that the top ranked authority and hub values ranked as ones. We don't have full packet captures for the ISP originated datasets, so we don't really know to what extent the traffic was malicious, but the ranks are distributed similarly. This is not the case for the honeypot case, where a larger quantity of flows do have significant hub values when compared to the authority values.

[4] http://packetstormsecurity.nl/0008-exploits/statdx.c

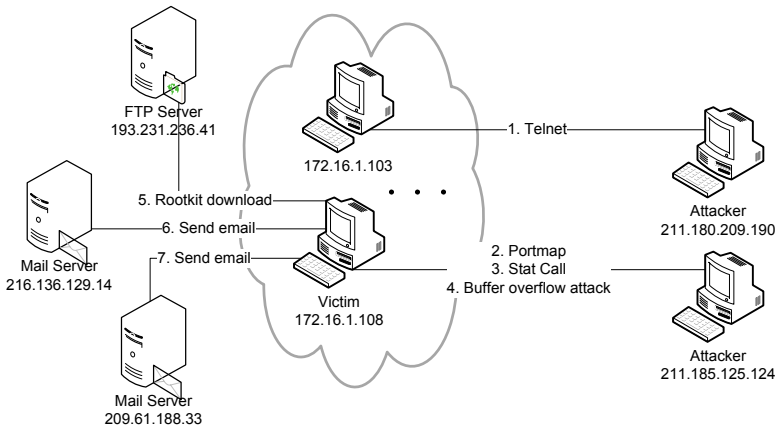


Fig. 3: Attack scenario

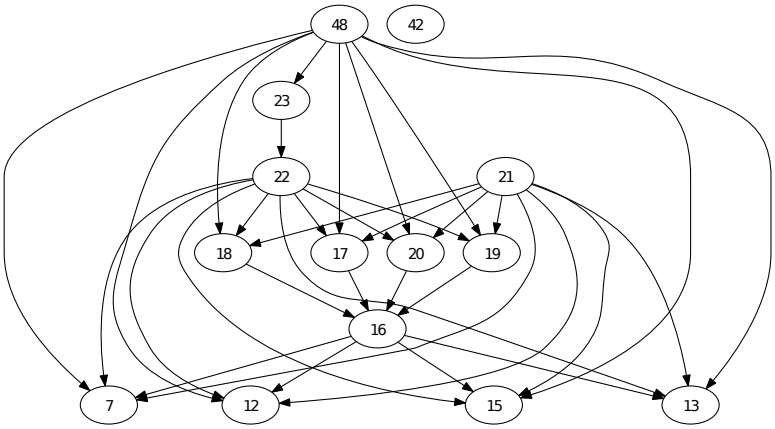
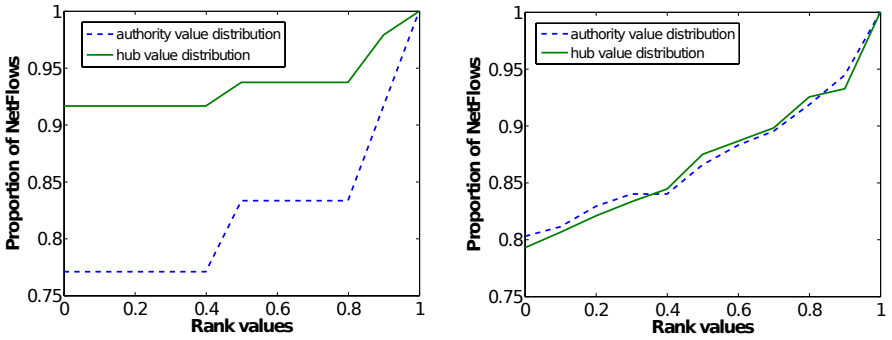


Fig. 4: NetFlow dependencies

Table 1: Attack activities

step	flowID	attack name	src IP: port	dst IP: port	packet#	bytes#	authority score	hub score
1	42	Telnet	211.180.229.190:3329	172.16.1.103:23	40	2856	0	0
2	12	Portmap	211.185.125.124:790	172.16.1.108:111	4	336	1	0
3	13	Stat call	211.185.125.124:791	172.16.1.108:931	6	6708	1	0
4	15	Buffer overflow	211.185.125.124:4450	172.16.1.108:39168	168	15642	1	0
5	16	Rootkit download	172.16.1.108:1026	193.231.236.41:21	74	6318	0	0.48
6	21	Send email	172.16.1.108:1028	216.136.129.14:25	48	5562	0	0.89
7	22	Send email	172.16.1.108:1029	209.61.188.33:25	40	3680	0	0.89



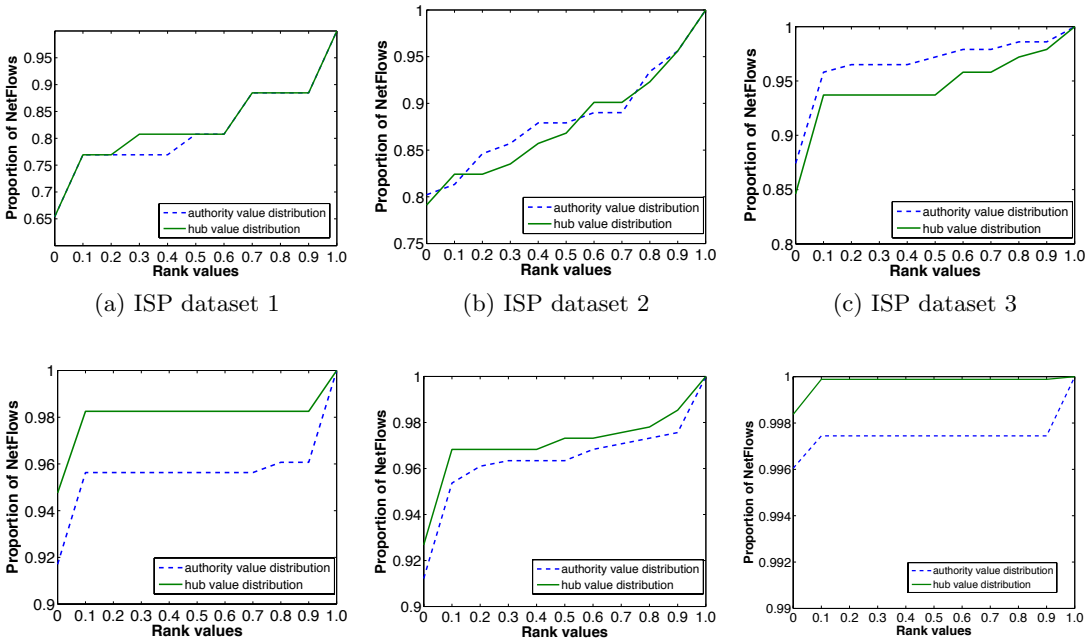
(a) honey pot attack scenario (b) ISP dataset

Fig. 5: NetFlow ranks cumulative histogram

In order to investigate the distribution of HITS ranking scores, we collected 6 data sets, each of which spans a time window of 1 minute. Figure 7 illustrates the proportion of zero ranked flow records, we refer zero ranked flow as flows with both zero authority value and hub value. The experiment shows that as the number of NetFlow records increases, the proportion of zero ranked flow records also increases, thus the number of non-zero ranked flow records stays within a manageable size for human interpretation. This shed a light on using our technique as a forensic tool for mining essential activities out of large scale of raw data.

Table 2: Flow Statistics

	attack scenario	ISP ds 0	ISP ds 1	ISP ds 2	ISP ds 3	ISP ds 4	ISP ds 5	ISP ds 6
Total Flows	48	658.1k	26	91	141	229	410	11.3k
Total bytes	1.2M	898.8M	141M	378.6M	369.6M	145.3M	3924M	124.3M
Total packets	2.3k	2.0 M	162k	468k	477.3k	216.3k	5.29M	214.8k
avg bytes/sec	19.0k	3.0M	2.35M	6.3M	6.15M	2.4M	65.4M	2.1M
avg packet/sec	0.036	6.8k	2.7k	7.8k	8.0k	3.6k	88.1k	3.6k
Destination Ports #	14	63879	21	72	106	140	313	8.9k
Distinct Source IP #	14	69310	16	58	91	154	243	1.6k
Distinct Destination IP #	14	70946	15	57	94	136	232	3.1k
Average Packets/Flow	48.3	3.04	5.8k	5.0k	3.3k	948.7	12.9k	19.0
Average Bytes/Flow	25.6k	1.37k	5.0M	4.0m	2.6m	637.4k	9.60M	11.0m



(a) ISP dataset 1 (b) ISP dataset 2 (c) ISP dataset 3

(d) ISP dataset 4 (e) ISP dataset 5 (f) ISP dataset 6

Fig. 6: Rank score distributions

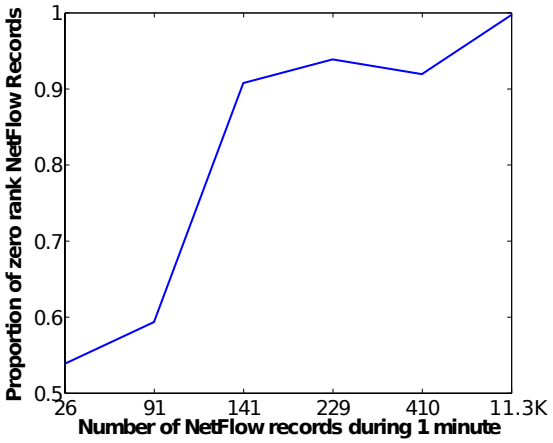


Fig. 7: Proportion of zero rank NetFlow records distributions

7 Related Work

Discovering dependencies among network traffic has been extensively studied previously in the network management community [10, 15, 11, 1, 4, 8, 9, 2, 5]. Our approach differs from the previous study mainly in that previous study focuses on discovering host level and application level dependencies in order to facilitate fault allocation, reconfiguration planning, and anomaly detection, while our objective is to reveal causal dependencies among NetFlows to detect abnormal intense activities online as well as the critical hosts that most other hosts require to function properly.

Previous works [9] [2] [5] [15] on dependencies discovery applied various statistics and probability techniques to reduce the false dependencies. These works are complementary to our model. In our current model, we have chosen a simple yet efficient model due to its better online performance to deal with the large amount of NetFlow records.

Sawilla [16] ranked attack graphs using pagerank [3], S Wang [17] ranked NetFlow record with pagerank. While both are link analysis algorithms, pagerank ranks nodes on a dependency graph based on their in-degree and HITS ranking also reflect out-degree as hub values. Considering out-degree of NetFlow records helps to reveal flows with destination address that are targets of large amount of network traffic, the ultimate goal of a series of attack activities for example.

8 Conclusion and Future Work

We have addressed in this paper a new method to detect relevant IP flow records. Our approach leverages the HITS algorithm to search for relevant nodes in dependency graphs. The dependency graphs are build by taking into account the potential causality among several flow records. Albeit simple, such a model can capture causal dependencies, where for instance one flow is the trigger for a large set of follow up network activity. We have applied our method on several datasets. The first dataset concerned a publicly available network capture from a forensic challenge and our method correctly identified the relevant malicious IP flows. We have also validated our method on a large IP flow capture from an ISP border router in order to assess its limits in terms of data volume.

We plan to extend our quantitative analysis for a larger set of different traffic scenarios, like for instance botnet detection. The current problem that we face is related to the baseline of an analysis method. Since our approach works on captured netflow records, we have no real evidence (complete packet captur) to compare with. The usage of full packet capturing is for legal reasons impossible. We are also investigating the potential application to a larger class of security events that include firewall logs, IPS data and syslog events within a larger context of event correlation and root cause analysis.

References

1. MK Aguilera, JC Mogul, JL Wiener, P Reynolds, and A Muthitacharoen. Performance debugging for distributed systems of black boxes. *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 74–89, 2003.
2. P Barham, R Black, M Goldszmidt, R Isaacs, J MacCormick, R Mortier, and A Simma. Constellation: automated discovery of service and host dependencies in networked systems. *TechReport, MSR-TR-2008-67*, 2008.
3. S Brin and L Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.
4. M Chen, A Accardi, E Kiciman, and J Lloyd. Path-based failure and evolution management. *NSDI'04*, Jan 2004.
5. X Chen, M Zhang, ZM Mao, and P Bahl. Automating network application dependency discovery: Experiences, limitations, and new solutions. *Proceedings of OSDI*, 2008.
6. Internet Engineering Task Force(IETF). Ip flow information export (ipfix). `http://www.ietf.org/dyn/wg/charter/ipfix-charter.html`, March 2010.
7. Network Working Group. Rpc: Remote procedure call protocol specification version 2. `http://tools.ietf.org/html/rfc5531`, March 2010.
8. M Iliofotou, P Pappu, M Faloutsos, M Mitzenmacher, S Singh, and G Varghese. Network monitoring using traffic dispersion graphs (tdgs). *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 315–320, 2007.
9. L Jian-Guang, F Qiang, and J Yi WANG. Mining dependency in distributed systems through unstructured logs analysis. *research.microsoft.com*.
10. S Kandula, R Chandra, and D Katabi. What's going on?: learning communication rules in edge networks. *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, pages 87–98, 2008.
11. J Kannan, J Jung, V Paxson, and CE Koksal. Semi-automated discovery of application session structure. *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 119–132, 2006.
12. Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM*, 46(5), Sep 1999.
13. AY Ng, AX Zheng, and MI Jordan. Link analysis, eigenvectors and stability. *International Joint Conference on Artificial Intelligence*, 17(1):903–910, 2001.
14. The Honeynet Project. Scan18. `http://old.honeynet.org/scans/scan18/`, March 2010.
15. P Reynolds, JL Wiener, JC Mogul, MK Aguilera, and A Vahdat. Wap5: black-box performance debugging for wide-area systems. *Proceedings of the 15th international conference on World Wide Web*, pages 347–356, 2006.
16. R Sawilla and X Ou. Identifying critical attack assets in dependency attack gaphs. *Proceedings of the 13th European Symposium on Research in Computer Security*, 13:18–34, 2008.
17. S Wang, R State, M Ourdane, and T Engel. Mining netflow records for critical network activities. *Proceedings of the 6th International Wireless Communications & Mobile Computing Conference*, 2010.