Attribution and Aggregation of Network Flows for Security Analysis

Annarita Giani, Ian Gregorio De Souza, Vincent Berk, George Cybenko
Thayer School of Engineering
Dartmouth College
Hanover, NH
{*agiani, idesouza, vberk, gvc*}*@dartmouth.edu*

Abstract

This paper describes a network flow analyzer that is capable of attribution and aggregation of different flows into single activity events for the purposes of identifying suspicious and illegitimate behaviors. Flows are correlated with security events using the Process Query System (PQS) infrastructure. We show results from initial experiments and describe plans for extending the effort. The correlation of networks flows with security events appears to have high potential for aggregating disparate network and host activity and for classifying network activity as either benign or suspicious.

1 Introduction

A flow sensor emits observations as detailed flows. The system generates a very large continuous quantity of data that cannot be processed by humans practically. A skilled security analyst that tries to analyze the flow characteristics may recognize an illegitimate connection but is likely to miss some malicious traffic. The fact that a flow is unjustifiable can be related to the flow breaking some criterion described in a policy file or to the fact that intrusion detection systems (IDS) or other sensors had evidence classifying the flow as illegal. For example, if we notice that a flow contains an FTP session and the initiator of the connection is a host that triggered a Snort sensor, the flow must be categorized as malicious. A system administrator might miss this correlation and report the connection as legitimate.

We have designed and implemented an approach to correlating network flows with security events, such as those generated by IDS, for the purposes of aggregating and attributing flows. This capability achieves data compression and provides insights to the analyst about whether flows are benign or suspicious. Our temporal and multi-sensor correlation system is based on a Process Query System which is designed to correlate different data based on the process models that it runs. Models serve to correlate data coming from flow anomaly detectors and other security tools.

Process Query Systems (PQS) [2, 7] are software systems that allow users to interact with multiple data sources, such as traditional databases and real-time sensor feeds, in a new and powerful way. In traditional databases, users specify queries expressed as constraints on the field values of records stored in a database or data recorded by sensors, as allowed by SQL and its variants for streaming data. By contrast, PQS allows users to define processes and to make queries against databases and real-time sensor data feeds by submitting those process definitions. A PQS parses the process description and performs sequences of queries against available data resources, searching for evidence that instances of the specified process or processes exist.

The strength of the system relies on the fact that the same PQS engine can be easily adapted to different processes. PQS has been successfully applied to ground vehicle tracking [6], social networks [4], and plume source detection [10]. For each of the preceding applications, the observations gathered are, respectively: positions as recorded by sensors, evidence of particular activity in the social network (initiator of a conversation, broker, etc.) and concentrations of a particular gas at a fixed location.

PQS is also a very powerful and efficient tool to perform network security monitoring. In this framework the system collects data from many different sensors and implements multilevel models that evaluate the data. As a result it returns conclusions that can be very specific or more general depending on which PQS tier is used. Previous implementations of PQS models [8, 9, 1] have detected malicious hacker behaviors, insider threat behaviors, network failures, worms, viruses and covert channels. In this implementation of the system we want to disambiguate between benign and malicious behavior in terms of flows. We want to develop a framework for using flow attribution and flow aggregation as part of se-

curity analysis. *Flow attribution* consists of detecting logs that can explain a flow. The final goal is attributing the flow to a person but intermediate steps are a required part of the attribution process. *Flow aggregation* means recognizing that different flows, apparently totally unrelated, nevertheless belong to the same broader event. Single flows are defined as components and groups of related flows are defined as events. Examples are an FTP connection followed by data transfer or surfing the web and connecting to different web pages following links present in the pages.

Flow attribution has been investigated mostly with the purpose of enhancing quality of service [5, 3] while the present work is devoted to security uses of network flows.

The paper is organized in the following way. Section 2 describes the flow sensor together with a description of the network in which it was deployed. Section 3 presents some of the other sensors used in our analysis. Section 4 shows results of an experiment that we ran and anticipates other experiments. Section 5 gives a brief overview about how to measure improvements in this framework.

2 Flow Sensor Description

The flow sensor that we built was deployed on an unsecured production network [1]. Although the network is small compared to an actual enterprise-class network, it features a wide range of systems and servers that are typical of larger organizations. Behind the firewall there are 64 addresses available (.192/26) which are split between a *Workstations* and a *Demilitarized Zone* server network, both (/27). The uplink connects directly to the internet and all addresses are globally routable. The Workstation network features several Windows XP clients, Linux 2.4 and 2.6 systems, several Solaris workstations, and a Solaris 9 server to which multiple thin-clients are connected. All these systems are used daily by researchers and students, and so the traffic on this network is typical for a normal operating organization where users browse the web, print documents, and download files during business hours.

Our flow sensor is based on the libpcap interface for packet capturing. Traffic flows was collected for a period of two months. Packets with the same source IP address, destination IP address, source port and destination port and protocol were aggregate in a single flow which remain active until no other packets with the same characteristic arrive for a period of 5 minute.. Each flow is characterized by fields that are used to infer quantities given as observations to the PQS models. These fields are:

1. Timestamp of the first packet.

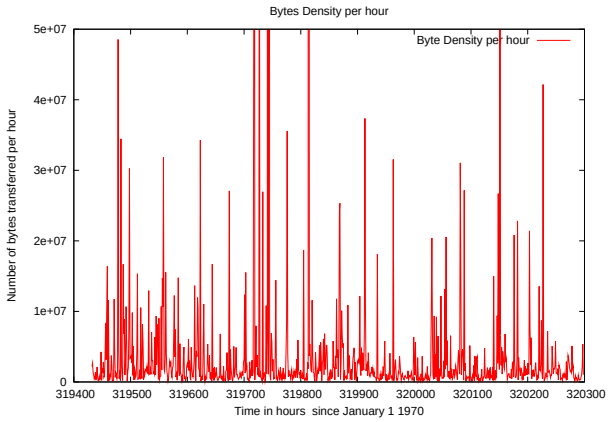

Figure 1: Bytes Density per hour from June 09 2006 04:00:00 to July 16 16:00:00

2. Timestamp of the last packet.
3. Number of packets from the source to the destination
4. Number of packets from the source to the destination
5. Number of bytes from the source to the destination
6. Number of bytes from the destination to the source
7. The IP address of the source machine
8. The IP address of the destination machine
9. The protocol
10. The source port
11. The destination port
12. An array containing the delays, in microseconds, between two consecutive packets
13. An array containing the number of bytes in each packets.

These quantities allow us to infer interesting statistics on the network traffic on our network. Figure 1 and Figure 2 show the density of flows and bytes transferred per hour respectively. We notice for example that the number of flows per hour is mostly between 50 and 100 with some sporadic spikes.

The delays between packets and the distribution of bytes in each flow embed information that can be used to investigate situations of covert channel embedded in inter-packets delays and also provide statistics on the distribution of load in each flow which will be relevant to the development of higher-level security tools.

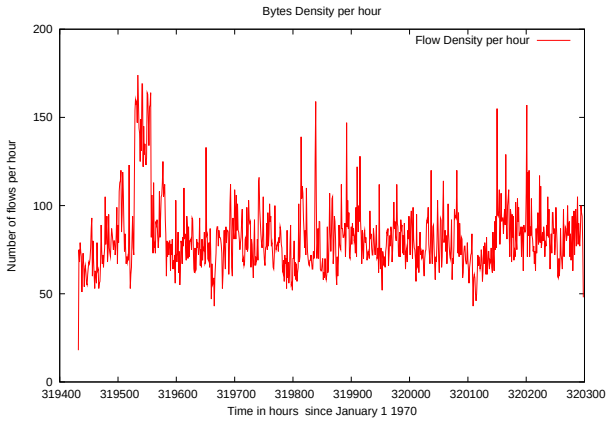

Figure 2: Flow Density per hour from June 09 2006 04:00:00 to July 16 16:00:00

3 Host and network sensors

PQSnet is our implementation of a PQS to the domain of computer security. Since the beginning of the PQSnet project three years ago, we have used a number of sensors based on common host and network security applications.

The *Snort* and *Dragon sensors* are both signature-matching Intrusion Detection Systems (IDS). The sensor's observations give information on source and destination IPs, ports, and which rule or signature was violated.

The *Web log sensor* reports information from Apache, IIS and SSL error logs. Observations from this sensor contain information about suspicious URLs that have succeeded or failed, as well as web server errors.

The *Samhain sensor* reports alerts from the Samhain host file integrity monitor. Samhain frequently checks critical system files on a host for additions, modifications and deletions. Any changes are immediately logged locally or reported to a remote log server. Samhain sensor observations include timestamps of changes, filenames, violation type, and changes in the system kernel. Observations from this sensor allows correlation of network and host based activity.

The *NetFlow sensor* classifies a communication between two or more machines in terms of volume of data transfer, length of the transmission and number of packets involved. The sensor monitors network packets and aggregates them in groups of the same source IP, destination IP, source port, and destination port. As packets in the same flow arrive, the number of bytes of new packets are recorded. The flow is considered closed if transmission stops for the duration of a specified time threshold.

Volume	Packets	Duration	Balance	%
0: 0 1: 1-128b Tiny 2: 128b-1Kb Small 3: 1Kb-100Kb Med. 4: $>$ 100Kb large	1: one packet 2: two packets 3: 3-9 4: 10-99 5: 100-999 6: $>$ 1000	0: $<$ 1 s Zero 1: 1-10 s Very Small 2: 10-100 s Small 3: 100-1000 s Medium 4: 1000-10000 s Large 5: 10000-100000 s Very Large 6: $>$ 100000 s Very Very Large	1: IN 2: OUT 3: PAR	N

Table 1: NetFlow sensor classifications

At that point, the number of packets and total duration of the flow is computed. These quantities are then classified as shown in Table 1. This is the first version of network flow sensor that we built. We still use it since it is useful to get observations that feed models like *large upload, low and slow download*, that are instantiated according to the volume, number of packets, duration and balance. Percentage is the actual percentage above 50 of data movement in the direction of the balance field.

4 Investigated scenarios

4.1 Flow attribution

4.1.1 Packets in a flow triggered IDS alerts

The PQS instantiates models based on observations coming from the flow and the Snort sensors. An example of a track formed is shown in Table 2. The columns represent the timestamp, the sensor type (F = Flow Sensor and S = Snort), the IP address of the initiator of the communication, the destination IP address, and the protocol respectively. This particular track shows that a single flow between two hosts triggered many Snort alerts. The numbers next to the Snort sensor type represent the particular Snort rule that was violated. Snort rule 1560 (*WEB-MISC /doc/access*) generates an alert when an attempt is made to exploit a known vulnerability on a web server or a web application. Snort rule 1852 (*WEB-MISC robots.txt access Summary*) generates an alert when an attempt is made to access the 'robots.txt' file directly. Also we must notice the starting and ending timestamp of the flow. As we can see the flow ends but the model keeps correlating the flow with snort alerts generated after the flow finished.

4.2 Flow aggregation

The number of flows collected over a network can be very high and belong to disconnected activities like chatting or retrieving a web page. But flows apparently unrelated to one another might still belong to a single user

Timestamp	Sensor	src IP	dst IP	Proto
Jul 09 16:28:32	S1852	65.54.188.140	208.253.154.195	TCP
Jul 09 16:29:35	S1852	65.54.188.140	208.253.154.195	TCP
Jul 09 16:44:44	S1560	65.54.188.140	208.253.154.195	TCP
Jul 09 18:26:08	S1560	65.54.188.140	208.253.154.195	TCP
Jul 09 21:05:03	S1852	65.54.188.140	208.253.154.195	TCP
Jul 09 22:31:08	S1852	65.54.188.140	208.253.154.195	TCP
Jul 09 22:31:08	S1560	65.54.188.140	208.253.154.195	TCP
Jul 10 02:45:19	S1852	65.54.188.140	208.253.154.195	TCP
Jul 10 02:45:23	S1852	65.54.188.140	208.253.154.195	TCP
Jul 10 09:21:15	S1852	65.54.188.140	208.253.154.195	TCP
Jul 10 14:33:43	S1852	65.54.188.140	208.253.154.195	TCP
Jul 10 17:54:54	S1852	65.54.188.140	208.253.154.195	TCP
Jul 10 22:07:02	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 01:38:09	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 04:05:54	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 04:20:00	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 04:20:00	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 11:07:12	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 11:56:12	S1852	65.54.188.140	208.253.154.195	TCP
Jul 11 17:16:59	S1852	65.54.188.140	208.253.154.195	TCP
S Jul 10 02:30:27 E Jul 10 23:55:56	F	65.54.188.140	208.253.154.195	TCP

Table 2: A sample track of correlated IDS and Flow events

event or action. Our goal is to correlate all these flows to a single user activity.

4.2.1 Surfing the web

The full retrieval of a web page is a classical example of this situation. The first flow represents a DNS query to retrieve the IP address of the web server. An HTTP query to the web server generates a second flow. At this point many other flows may be generated (for example, as images download).

Another more challenging example is the following. Users often start connecting to a web page, and then follow links present in the page they visit. Since each page might have different flow characteristics this activity results in many different flows. But it is reasonable to think about these flows as part of the same event. Time stamps can be used to discriminate different instantiations of the same event. The challenge is to identify unrelated flows as part of a single event.

Building models for web browsing aggregation requires retrieving flows with destination port 80, parsing the web pages and following the links. If an IP address or URL linked by the web page is present in one of the flows it might be that the two requests are connected.

5 Improvements metrics

While developing a framework for flow tracking it is important to measure the success of our system. We must therefore find ways to quantify progress. One way of measuring improvement in our method is checking the size of the group of actions to which a given observable or flow may be attributed. If the size as well as the ambiguity reduces the methodology becomes more and more precise. Each attribution or aggregation must be supported by a level of confidence quantified with a number indicating how certain we are of the conclusions. Also, we can improve the system by increasing the number and type of observations that it can aggregate into a set of basic components.

6 Conclusion and Future Work

Process Query Systems are powerful tools to perform sensor observation correlation. We applied the system to correlate flow observation with other IDS observations successfully. We plan to run experiments to perform flow aggregation in order to identify flows belonging to a single user event. We believe that this would lead to significant improvements in security analysis.

7 Acknowledgments

Supported under ARDA P2INGS Award No. F30602-03-C-0248. Points of view in this document are those of the author(s) and do not necessarily represent the official position of ARDA.

References

[1] V. Berk and N. Fox. Process Query Systems for Network Security Monitoring. *Proceedings of the SPIE*, 5778, April 2005.

[2] V. H. Berk, W. W. Chung, V. Crespi, G. Cybenko, R. Gray, D. Hernando, G. Jiang, H. Li, and Y. Sheng. Process query systems for surveillance and awareness. *Proceedings of Systemics, Cybernetics and Informatics (SCI2003,* July 2003.

[3] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt. Flow Aggregation for Enhanced TCP over wide-area wireless. pages 1754–1764, 2003.

[4] W. Chung, R. Savell, J. P. Schtt, and G. V. Cybenko. Identifying and Tracking Dynamic Processes in Social Networks. *Proceedings of the SPIE*, 6201, April 2006.

[5] J. A. Cobb. Preserving Quality of Service Guarantees in spite of Flow Aggregation. *IEEE/ACM Trans. Netw.*, 10(1):43–53, 2002.

[6] V. Crespi, W. Chung, and A. B. Jordan. Decentralized Sensing and Tracking for UAV Scheduling. *Proceedings of the SPIE*, 5403, April 2004.

[7] G. Cybenko, V. H. Berk, V. Crespi, R. S. Gray, and G. Jiang. An Overview of Process Query Systems. *Proceedings of the SPIE*, 5403, April 2003.

[8] I. G. de Souza, V. H. Berk, A. Giani, G. Bakos, M. Bates, G. Cybenko, and D. Madory. Detection of Complex Cyber Attacks. *Proceedings of the SPIE*, 6201, April 2006.

[9] A. Giani, V. Berk, and G. Cybenko. Covert Channel Detection Using Process Query Systems. *FloCon*, 2005.

[10] G. Nofsinger and G. Cybenko. Distributed Chemical Plume Process Detection. *IEEE MILCOM*, 2005.