Visualizing Host Traffic through Graphs

Eduard Glatz
Computer Engineering and Networks Laboratory
ETH Zurich
eglatz@tik.ee.ethz.ch

ABSTRACT
Gaining an overview of host activities is hard when a host is busily exchanging hundreds or thousands of flows over a network. This makes investigating traffic of a suspicious host a tedious task for a security analyst. We propose a novel host traffic visualization technique that reduces this cognitive burden by i) representing traffic through an annotated k-partite graph reflecting familiar Berkeley socket model semantics, ii) employing a host role summarization for effective removal of ephemeral traffic features, and iii) providing classification and filtering techniques for unwanted traffic, which are important for identifying the functional role of port numbers and for visualization. We present the open-source tool *HAPviewer* and demonstrate how it can visualize a large number of flows through a compact and easily interpretable graph.

Categories and Subject Descriptors
C.2 [Computer-Communication Networks]: General—Security and Protection; H.5 [Information Interfaces and Presentation]: General

General Terms
Measurement, Security.

Keywords
Network, security, information visualization.

1. INTRODUCTION
Inspecting host activities is a crucial step in the process of security incidence analysis. But, looking up flow lists is a tedious task when they comprise hundreds or even thousands of flows. As a consequence this task puts a considerable cognitive burden onto a security analyst degrading his productivity.

Information visualization is a popular approach to bring masses of data into a form which is easier for us to read. One form of visualization is the use of graphs where data is represented through nodes and links, which both can be annotated with additional information. Graphs express relations that otherwise remain hidden. However, applied to transactional data such as network traffic flow data, a graph representation can lead to a cluttered image that is hard to read. We illustrate this by looking at a simple example visualizing four transactions (see Figure 1a), each comprising four attributes w, x, y, z with indices representing distinct attribute values. Nodes correspond to attribute values and links denote their co-occurrence. The resulting picture is hard to read due to the random node placement and the many crossing links.

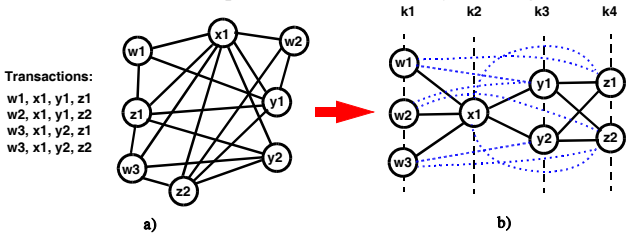

Figure 1. Example visualizing four transactions through a graph in two different ways.

A better representation requires us to decide which relationships (links) are most interesting to illustrate and how to place nodes in a well-organized way. We propose the use of a k-partite graph with a column-wise organization of node placement and the pruning of less important links. We order partitions such that interesting relationships only occur between neighboring partitions resulting in a near planar graph. Figure 1b redraws our example transaction set accordingly with pruned links shown as dotted lines.

Extending the attribute list to a more complete set of attributes leads us to our *first contribution*, namely the *host application profile (HAP) graphlet* (see Figure 2). It represents host traffic in a familiar way based on the key attributes of the Berkeley socket model. We describe the details of the HAP graphlet definition, such as how we make use of node and link annotations, in Section 2.1.

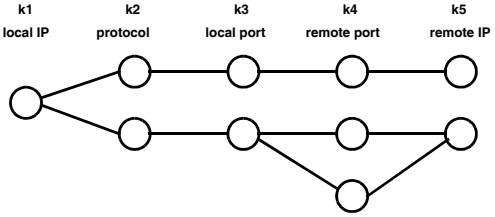

Figure 2. Host application profile (HAP) graphlet visualizing host traffic through a constrained 5-partite graph.

Still, we face the problem that our graph contains too many details distracting us from the big picture when we visualize very long flow lists. We address this problem by proposing a summarization technique as a *second contribution* that aggregates functionally related flows into host roles that can be visualized by an aggregated form of our graph. Figure 3 shows a simple example of how a server role can be summarized and, thus, the graph representation can be simplified. In the summarized graph (Figure 3b) the

VizSec '10, September 14, 2010, Ottawa, Ontario, Canada.

node annotation in k4 indicates the number of connections involving service port 80 and in k5 the number of clients contacting local IP as a server. In Section 2.2 we introduce a set of role definitions for which we support role summarization.

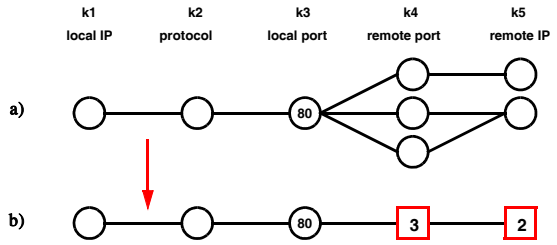

Figure 3. Example of a server role summarization that aggregates functionally related flows into a simpler graph.

Our *third contribution* focuses on *visual cues* about the nature of traffic. We distinguish productive and unproductive flows by using a simple heuristic. This heuristic assumes that productive traffic involves bidirectional communication between a host pair. If a flow sent by a host A stays unanswered by destination host B then this likely is a suspicious flow needing further investigation. We visualize flow directionality by arrows and color variation. We introduce the details of this classification in section 2.3.

Our visualization approach is implemented in an open-source tool which we call *HAPviewer*. We demonstrate its usefulness in Section 3 with selected case studies showing how an analyst can make use of it. In Section 4 we discuss how we achieve an effective visualization. In Section 5 we review related work, followed by future directions in Section 6. Section 7 concludes this paper.

2. HAPviewer

In this section we describe the host traffic representation in detail, define host roles and their summarization and explain how traffic is classified into productive/unproductive flows.

2.1 Host Traffic Representation

A basic description of a Berkeley socket based network connection consists of a 5-tuple containing the attributes {source IP address, destination IP address, protocol, source port, destination port}. We define a 5-partite graph which puts end host into the first and last partitions and assigns the middle partitions to the port and protocol attributes (see Figure 2 and 3 for examples). This yields a representation which puts layer 3 connectivity at both ends and layer 4 connectivity in between, making it visually easy to grasp. Node annotations denote values of the 5-tuple while link annotations display additional attributes such as byte counts (packet counts in parenthesis) between partitions k3/k4, and flow counts (average packet per flow ratio in parenthesis) between partitions k4/k5. We make use of the dot algorithm to layout the graph [2]. We name this traffic representation the *host application profile (HAP) graphlet* as it illustrates used protocols and the functional role of local and remote port numbers in the context of application processes running on the end hosts. A HAP graphlet groups flows per application in a way that host roles are apparent. We introduce a set of basic roles in the next section.

2.2 Host Role Summarization

We define a set of host roles as shown in Figure 4. Frequently occurring roles are a/b for servers, c/d for clients and e/f/g for peers. Role d summarizes a set of client roles with equal remote port numbers, e.g. port 80/tcp for multiple web server access by a client. P2P roles involve flows having both port numbers above 1024, or which are exchanged with a remote host communicating over udp and tcp simultaneously. Furthermore, any client roles using service port numbers above 1024 are summarized into P2P role g. Finally, we note that summarization is performed in alphabetical order of role definitions to break a tie.

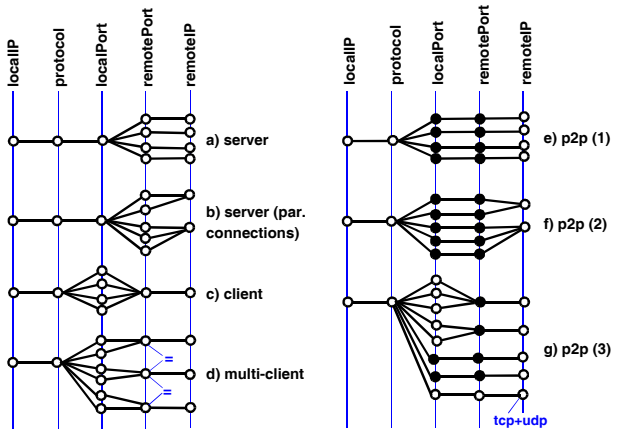

Figure 4. Host roles definitions based on characteristic connection structures of server, client and P2P applications. Filled nodes represent port numbers above 1024.

Each role can be characterized by its subgraph as part of a complete HAP graphlet. Host role summarization involves the replacement of role subgraphs by summary subgraphs that consist of a single path through all partitions involved. Figure 3 shows an example of a server role summarization. To distinguish summary subgraphs from single flows we use squared nodes instead of the regular ellipses. We provide statistics about summarized connections and the number of remote hosts involved as summary node annotations. Figure 5 illustrates the effectiveness of our approach by summarizing 1082 flows into a compact graph.

2.3 Flow Classification and Filtering

Regular applications use bidirectional communications to acknowledge received data. This is done either on the transport layer for TCP connections or on the application layer for UDP transfers.

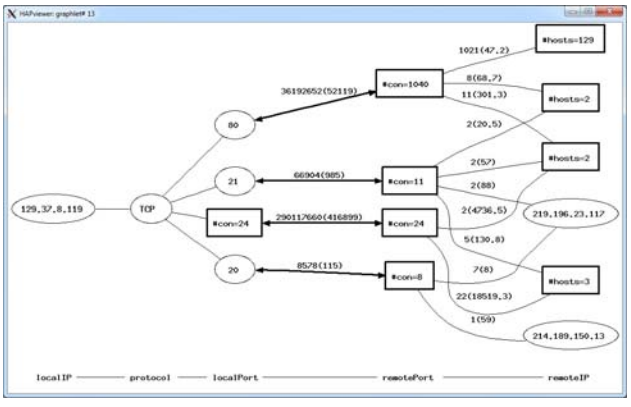

Figure 5. Example of a HAP graphlet using host role summarization to display 1083 flows in a compact graph.

On the other hand, popular flow formats such as Cisco's NetFlow or IPFIX define unidirectional flows. This makes it hard for an

analyst to align flows in such a way that he immediately recognizes unidirectional communication which often is a trademark of suspicious activity. To alleviate this deficiency we include a flow classification as part of data preprocessing. By matching flows in opposite direction which have identical attribute values of the 5-tuple (after exchanging local/remote attributes) we create bidirectional flows. Any flow not having a buddy in opposite direction is classified as unidirectional. This classification is used to provide *visual cues* in form of thin and thick arrowed lines in different colors when visualizing the HAP graphlet. A thick link between partitions k3/k4 having arrows at both ends marks bidirectional communication, while links with a single arrow at one end represent unidirectional communication. To distinguish sporadically failing connections from other types of unidirectional traffic we use a green color to mark unidirectional flows exchanged between a host pair otherwise having bidirectional communication. Any remaining unidirectional flows are marked in red (see Figure 6 for an example). Of course this classifications can be used for flow filtering as well, e.g. for removal of all unidirectional flows from a displayed HAP graphlet. Ideally, bidirectional flow matching needs a warm-up and cool-down phase embracing the observation interval to support proper flow classification across interval borders. In a first implementation of HAPviewer we do not support this feature.

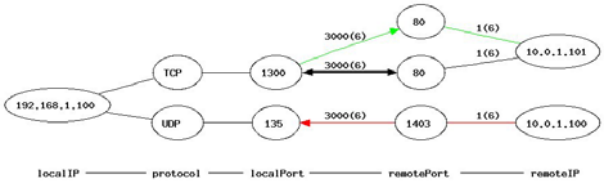

Figure 6. Example showing flows marked as outgoing unidirectional, bidirectional and incoming unidirectional.

3. Case Studies

In this section we illustrate the usefulness of our tool *HAPviewer* through selected case studies.

3.1 Situational Awareness

Scan traffic is a plague that will not go away in near future. However, it is also a source of information for a security analyst as scanning usually is a precursor of attacks. Being aware of changes in popular scan destination ports, the frequency of scans hitting individual hosts and identifying scan sources are common analysis tasks. We use a case study in which an analyst explores a large flow data set with the goal to detect interesting hosts and analyze their behavior. We use real flow traffic traces as a data source[1].

HAPviewer provides a host browser that is useful to obtain a fast overview of host activities (see Figure 7). This host browse list shows the IP address, a host number as a reference, total and unidirectional flow counts, number of protocols used and the count of packets/bytes exchanged. All columns of this list can be sorted by their column title, e.g. for descending unidirectional flow counts which helps to create top-N lists. Figure 7 shows a group of hosts with neighboring IP addresses that all experience three unidirectional flows of the same size.

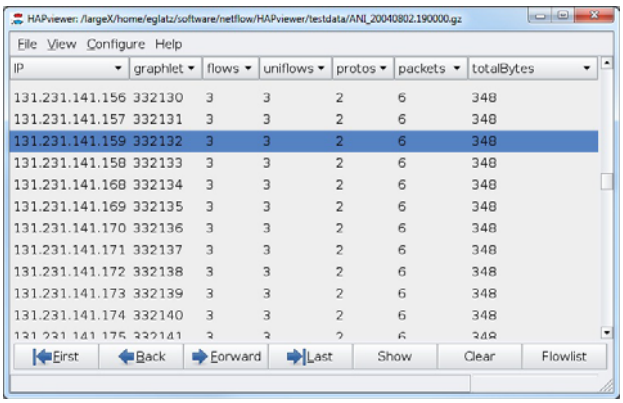

Figure 7. HAPviewer's host browse list showing an excerpt of example data.

This seems to be interesting and so we use button *Show* to display the HAP graphlet of the highlighted host #332132 having IP address 131.231.141.159 (see Figure 8).

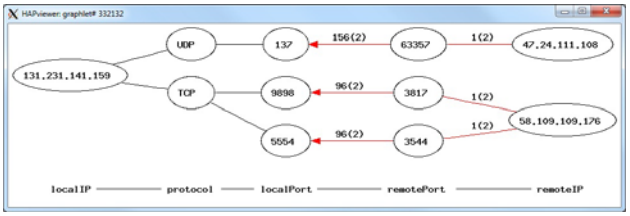

Figure 8. HAP graphlet of a host as a target of three unidirectional flows suspected to be scan traffic.

We immediately recognize two remote hosts as senders of unidirectional flows destined to ports 137/udp, 9898/tcp and 5554/tcp. All these ports are known to be popular targets for scan traffic, e.g. NETBIOS attacks on 137/udp, Dabber/Sasser worm activity on port 5554/tcp and port 9898/tcp probed as a potential backdoor left behind by prior Sasser worm infections [13].

Next, we investigate the two remote hosts to confirm their role as scan traffic sources. For this purpose the *goto IP address* search feature of HAPviewer comes in quite handy. All activity seen on remote host 47.24.111.108 during the observation interval of 10 min is a probing of 256 hosts on port 137/udp (see Figure 9). This looks like a scanner targeting a complete /24 subnet.

Figure 9. Example of a scanner probing port 137/udp on 256 hosts.

We confirm this by drilling down into the list of flows associated with this host. This list can be displayed in a separate window by clicking button *Flowlist* exposed by the graphlet window. Figure 10 shows a part of this flow list comprising a set of typical flow attributes including remote IP addresses and flow start times. We sort the list for ascending remote IP addresses and then notice that IP addresses in fact are consecutive.

[1] This and all other examples are taken from a 10 minute flow trace captured at the border of a medium sized ISP comprising 3031759 flows and 763450 local hosts. All IP addresses are anonymized in a prefix-preserving way for privacy reasons [3].

We confirm this by drilling down into the list of flows associated with this host. This list can be displayed in a separate window by clicking button *Flowlist* exposed by the graphlet window. Figure 10 shows a part of this flow list comprising a set of typical flow attributes including remote IP addresses and flow start times. We sort the list for ascending remote IP addresses and then notice that IP addresses in fact are consecutive.

direction	remoteIP	remPort	bytes	packets
--->	131.231.141.0	137	156	2
--->	131.231.141.1	137	156	2
--->	131.231.141.2	137	156	2
--->	131.231.141.3	137	156	2
--->	131.231.141.4	137	156	2
--->	131.231.141.5	137	156	2
--->	131.231.141.6	137	156	2
--->	131.231.141.7	137	156	2
--->	131.231.141.8	137	156	2
--->	131.231.141.9	137	156	2

Figure 10. Details of an example flow list sorted by remote IP address.

Now, we inspect the second remote host with IP address 58.109.109.176. This host again seems to be a scanner probing the ports 5554/tcp and 9898/tcp on many hosts (see Figure 11).

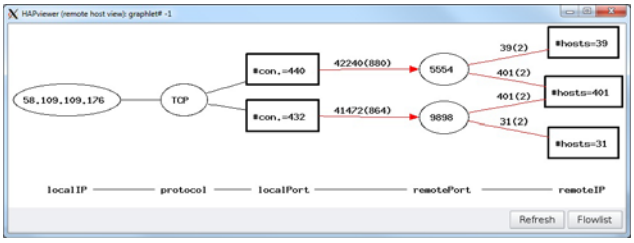

Figure 11. HAP graphlet showing a scan pattern involving 872 flows.

We note that the HAP graphlet easily shows us the scan pattern employed by this scanner although it involves a high number of 872 flows. A total of 401 hosts are scanned on both ports, a group of 39 hosts is scanned on port 5554/tcp exclusively and another group of 31 hosts receives scan flows on port 9898/tcp only. We determine the target IP ranges and the total scan activity time period by employing the sort function of the flow list associated with this host (not shown). We find that two /24 subnets are almost completely scanned and the scan activity lasted for 107.102 seconds. During this activity period groups of scan flows are issued using a high scan rate interrupted by random pauses ranging from 50 up to 300 ms. Furthermore, we notice that the complete scan activity period is within our observation interval indicating some obfuscation technique at work using an irregular time schedule.

3.2 Analysis of an IDS Alarm

Characterizing scanners is one task, investigating an alarm reported by an intrusion detection system (IDS) is another task. If a host has been successfully infected then it starts some form of malicious activity which might be well hidden within regular application traffic. Inspecting the communication pattern of a host is a key to unhide such malign communication. Very often IDS alarms trigger an in-depth investigation where an analyst gets the task assigned to confirm or deny the alarm raised.

Consequently, we have chosen a case study in which a particular host has been marked as suspicious and needs to be investigated in more detail. Figure 13 displays the HAP graphlet of the example host to be inspected involving 211 flows and 19583 packets. Again, we make use of real traffic traces. Firstly, we recognize two incoming scan flows destined to ports 135/tcp and 445/tcp which obviously are unrelated to host activities. These flows can be classified as scan traffic due to their packet counts of 1 and 2. These packet counts are too small to describe productive traffic as any established tcp connection needs at least 3 packets. Additionally, the packet size of 48 byte indicates missing payload. In a next step we notice two server roles, one on port 8069/udp and another on port 4662/tcp. Both server roles involve responding and non-responding clients. Particularly interesting is a group of 7 clients that act as servers and clients simultaneously using service port 4662/tcp. We note that this relationship can be seen in the HAP graphlet with ease, pinpointing the useful property of graphs to visualize multi-dimensional relationships. So far, we have identified several features typically associated with P2P applications: i) the presence of large groups of clients on tcp and udp using high port numbers, ii) some clients that also act as servers, iii) a significant number of non-responding clients. In fact, port number 4662 is known as a standard port of the eDonkey network. To confirm our findings we analyze how the large tcp traffic volume of 9163784 bytes exchanged over remote service port 4662/tcp is distributed across the remote peers.

remoteIP	remPort	bytes ▲	packets	flowstart	duration
46.169.219.10	4662	3929124	8144	19:01:26.423	749.247 s
175.71.64.7	4662	1388145	2517	19:01:26.871	975.754 s
45.167.237.127	4662	1227558	1853	19:01:26.423	794.825 s
47.74.31.192	4662	868069	1416	19:05:50.876	224.193 s
47.52.71.34	4662	593769	1423	19:02:46.085	310.489 s
207.6.174.254	4662	329918	678	19:00:29.592	80.903 s
91.68.69.151	4662	295375	503	19:01:31.478	207.304 s
91.156.30.84	4666	172665	372	19:06:14.557	213.890 s
44.95.92.151	4662	161521	346	19:01:28.733	200.513 s
46.201.221.113	4662	140754	249	19:01:43.958	253.506 s
44.95.92.151	4662	119846	222	19:05:01.258	93.780 s
221.163.25.106	4662	16670	45	19:07:36.094	73.472 s
211.117.191.63	4352	11934	104	19:01:10.875	5.248 s
47.55.214.99	4662	3657	28	19:05:56.814	138.448 s

Figure 12. P2P example flow list sorted by descending byte counts.

Figure 12 displays the flow list sorted by descending byte counts. We notice that 11 remote peers exhibit a byte volume of more than 100 KB with 5 of them exchanging more than 579 KB. This is a typical scenario for file swarming as employed by many P2P applications where files are downloaded in parts sup plied by different peers to leverage asymmetric down- and upload bandwidths offered to home users.

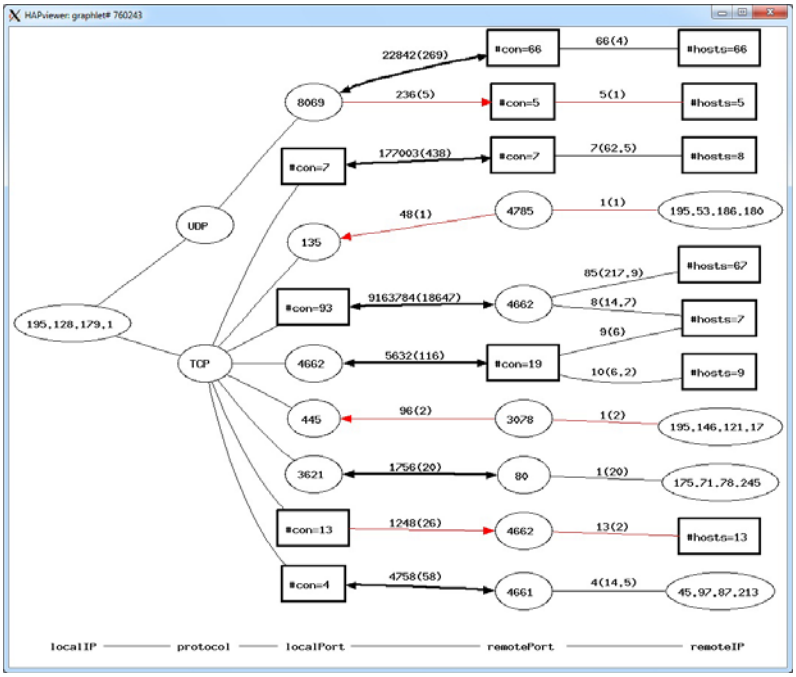

Figure 13. Example of a host displaying a complex P2P communication pattern.

Finally, we inspect a group of 7 flows being part of a P2P role. We notice that they use high ports and, therefore, likely belong to the running P2P application.

Now, we have almost inspected all flows exchanged but a single one using port 80. It is known hat P2P applications use port 80 for obfuscation and to bypass firewall filters. We check the remote host contacted on remote port 80 and see that it just communicates with our local host. We conclude that this remote host most likely is also a peer and not a server due to its behavior. This completes our analysis by concluding that this IDS alarm does not indicate a security incident but a P2P policy violation.

4. Discussion

Whenever a new approach to information visualization is presented the question is raised how useful it will be. Ultimately, a visualization is effective if it enables a user to improve sense-making from data in qualitative or quantitative terms. Information visualization literature has identified four different processes of insight that make visualization effective [11]. We discuss our approach in the context of these four key processes:

1. *Provide overview*: we provide an overview of all hosts through a host list that displays quantitative aggregates on a per–host level. This list can be sorted by any of the displayed host attributes just by clicking on the column titles. This helps an analyst to efficiently explore large host populations and identify interesting hosts. Furthermore, we plan to integrate our visualization into the NfSen analysis framework (see section 6 for details).
2. *Adjust*: this process involves the exploration of a dataset by adjusting the level of abstraction or the range of selection. We support this in three different ways: i) through enabling/disabling summarization individually per role type, ii) by drilling down into data through per-graphlet flow lists that can be sorted by any flow attribute, and iii) by extensive filtering options, e.g. by protocol and by flow directionality. This enables an analyst to switch forth and back between more abstract or detailed views of the data.
3. *Detect pattern*: our graphlet shows communication in a graphical way that stimulates structure recognition as part of natural human perception. This helps an analyst to gain a quick overview on what a host is doing and how similar it behaves compared with other hosts inspected.
4. *Match mental model*: we base our traffic representation on the Berkeley socket model as it has been learnt by any network professional. By using a simple and fixed arrangement of partitions in a way that conforms to a layered communication we provide a view which immediately gets familiar. This helps an analyst to quickly focus on essential details such as role types and service port numbers.

5. Related Work

We review how others use graphs to represent network traffic of individual hosts and host groups. Representing host traffic as a graph is not new. In [8] small k-partite graphs, called graphlets, are introduced to illustrate a novel behavioral application identification approach (BLINd Classification, *BLINC*). A predefined and fixed set of 12 graphlets with either 4 or 5 partitions are used to characterize typical communication structures associated with different application classes. A variation of the BLINC graphlet is proposed by [9] to infer fingerprints of end hosts for anomaly detection. These end host graphlets are constructed node by node under the constraint that only frequently occurring nodes are accepted. The tool *VisFlowConnect* [12] provides an overview of incoming and outgoing traffic of a complete network. It visualizes traffic between inside and outside hosts through a parallel axes view known in data mining. This view

arranges hosts in three partitions displayed as three vertical axes assigned to outside sending hosts, inside hosts and outside receiving hosts (from left-to-right). This makes suspicious unidirectional traffic visible in a unique way and gives a nice overview of a complete host group. *Traffic dispersion graphs (TDGs)* [7] visualize the social behavior of host groups, i.e. answer the question "who talks to whom". Nodes identify individual hosts and links represent a particular type of interaction an analyst is interested in. By use of application filtering a set of distinct characteristic graph patterns can be identified. *OverFlow* [4] visualizes networks of interest with nodes placed on concentric circles according to their hierarchical level. Connecting lines indicate communication, and help an analyst to spot unusual changes needing an in-depth inspection. *NFlowVis* [10] uses treemaps and force-directed graph layouts to effectively illustrate communication patterns of host groups being part of an attack. In [1] visualization of IPv6 data is discussed and how treemaps can be effectively employed to illustrate hierarchies.

Most related to our work is [9] in focusing on the traffic of individual hosts and capturing attribute relations. However, we differ in i) using a more intuitive graphlet that is favorable to visual interpretation, ii) summarizing flows in a way that retains information on all hosts involved, and iii) providing an open-source tool that is publicly available. This makes our tool useful to easily identify client, server and P2P roles, and for the investigation of specific security incidents.

6. Future Work

We plan to extend our tool in two directions:

- *Integration into existing tool sets*: we plan to integrate a library version of HAPviewer into the NfSen frontend of the NFDUMP netflow tools [6] to improve usability for security professionals [2]. This enables an analyst to quickly identify suspicious hosts and inspect them in detail through a clickable HAP graphlet [5]. We plan to do a user study with this integrated tool. Moreover, we will support common flow file formats such as defined by the IPFIX standard, the SiLK and NFDUMP tool sets.
- *Temporal flow selection*: we envision an additional view displaying the flow start time distribution. An analyst will be able to use time sliders to select groups of flows within a time window with variable start time and size leveraging temporal correlations induced by functionally related flows.

7. Conclusions

In this paper we introduce a host traffic visualization which helps an analyst to gain a quick and easily interpretable overview of host activities involving hundreds or thousands of flows. In detail, we support this claim by i) representing traffic through an annotated k-partite graph reflecting familiar Berkeley socket model semantics, ii) employing a host role summarization for effective removal of excessive traffic features, and iii) providing classification and filtering techniques for unwanted traffic, which are important for identifying the functional role of port numbers and for visualization.

We present the open-source tool *HAPviewer,* that efficiently processes several millions of flows and implements our visualization approach. We demonstrate its usefulness through selected case studies.

8. ACKNOWLEDGMENTS

The author is grateful to Xenofontas Dimitropoulos for his insightful suggestions and comments.

9. REFERENCES

[1] Barrera D, van Oorschot PC 2009. Security Visualization Tools and IPv6 Addresses. Workshop on Visualization for Cyber security, Atlantic City, NJ USA (VizSec 2009).

[2] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo. A Technique for Drawing Directed Graphs. IEEE Trans. Sofware Eng., 19(3):214–230, May 1993.

[3] Fan, J, Xu, J, Ammar, MH, and Moon, SB. 2004. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. Computer Networks 46, 2 (7 Oct. 04), 253-272, Elsevier.

[4] Glanfield J, Brooks S, Taylor T et al. 2009. OverFlow: An Overview Visualization for Network Analysis. Workshop on Visualization for Cyber Security (VizSec 2009).

[5] Fink, G., Endert, A. 2009. Visualizing Cyber Security: Usable Workspaces. Workshop on Visualization for Cyber Security (VizSec 2009).

[6] Haag, P. 2005. Watch your Flows with NfSen and NfDump. 50th RIPE Meeting (3 May 05, Stockholm).

[7] Iliofotou, M, Pappu, P, Faloutsos, M, Mitzenmacher, M, Singh, S, and Varghese, G. 2007. Network monitoring using traffic dispersion graphs (tdgs). In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement. ACM New York, NY.

[8] Karagiannis, T, Papagiannaki, K, and Faloutsos, M. 2005. BLINC: multilevel traffic classification in the dark. ACM SIGCOMM Computer Communication Review. ACM Press New York, NY.

[9] Karagiannis, T, Papagiannaki, K, Taft, N, and Faloutsos, M. 2007. Profiling the End Host. Passive and Active Measurement Conference (PAM), Louvain-la-neuve, Belgium, April 2007. SPRINGER-VERLAG.

[10] Mansmann F, Fischer F, Keim DA, North SC 2009. Visual support for analyzing network traffic and intrusion detection events using TreeMap and graph representations. New York, NY, USA: ACM.

[11] Yi, JS, Kang, Y, Stasko, JT, and Jacko, JA. 2008. Understanding and characterizing insights: how do people gain insights using information visualization? In Proceedings of the 2008 conference on BEyond time and errors: novel evaLuation methods for Information Visualization (Florence, Italy, 5 April 08). BELIV'08. ACM, New York, NY.

[12] Yin, X., Yurcik, W., Li, Y., Lakkaraju, K. and Abad, C. 2004. Visflowconnect: Providing security situational awareness by visualizing network traffic flows. Performance Computing and Communications, IEEE International Conference o, pages 601-607, 2004

[13] Xforce (IBM Internet Security Systems), Vulnerability ID 16244, Dabber worm detected. http://xforce.iss.net/xforce/xfdb/16244