HOSPITAL: Host and Network System Profiler and Internet Traffic Analyzer

Esam Sharafuddin, Nan Jiang, Yu Jin and Zhi-Li Zhang
University of Minnesota
{shara,njiang,yjin,zhzhang}@cs.umn.edu

***Abstract*—The ever-increasing complexity and diversity of the Internet pose several challenges to network operators and administrators and, in general, Internet users. More specifically, because of the diversity in applications and usage patterns; the prevalence of dynamic IP addresses and applications that do not conform to standard configuration (e.g. VoIP to bypass firewalls), monitoring and securing networks and end hosts is no longer a trivial task. In this paper, we propose *H*ost and netw*O*rk *S*ystem *P*rofiler and *I*nternet *T*raffic *A*na*L*ysis (HOSPITAL): a tool for the summarization, characterization of traffic and the troubleshooting of potential suspicious activities. HOSPITAL provides the network operator as well as the user with knowledge about applications, communicating parties, services required/provided, etc, at different levels of granularity (e.g. individual hosts, /24 blocks, a large enterprise, etc), all presented concisely with an easy to use web interface. Moreover, HOSPITAL is a light-weight self-contained tool that incurs little overhead with configuration and customization capabilities for users and developers.**

I. Introduction

In recent years, Internet related applications have become an indispensable component in everyone's daily life. People rely on VPN to work remotely, initialize telephone meeting with VoIP, play games and watch movies online, etc. The increasing number of applications and their inherent dependencies inevitably leads to much more complexity in the end system, making it an invisible "blackbox". It is hard nowadays, even for a savvy user, to tell "what is my computer doing right now?" or "what are these outgoing connections from my home network to the outside Internet?" Such uninterpretability of the end systems makes them vulnerable to potential attacks.

Traditional solutions for anomaly detection and troubleshooting often make use of traffic data which is readily provided by the routers, e.g., Cisco NetFlow data. For example, typical intrusion detection/prevention systems (IDS/IPS) [1], [2] identify network anomalies from such flow-level data. However, no matter how we place the router, flows among internal hosts are unobservable since they do not pass the router. In addition, due to the sheer volume of the traffic at the border routers, sampling is an inevitable solution to improve scalability. Both factors render such flow data a less accurate representation of the traffic in the network. Though such inaccuracy has little impact for detecting network-scale exploits or troubleshooting network-wide problems, it is intolerable for the "last-mile" diagnosis of host-level/subnet-level problems.

To the other extreme, a number of tools enable users to collect raw traffic data from either a computer or a subnet, such as tcpdump and wireshark. However, relying on users to mine meaningful information from such raw traffic data is unrealistic. Built on top of these traffic collection mechanisms, a variety of host-level tools, such as virus detection softwares and firewalls to detect malwares such as virus and worms, etc. Though being widely deployed to protect networks and end hosts, the design goal of these tools is to accurately and automatically detect anomalies. Therefore, they are general, require no user participation and hence provide little information to help understand host activities. As a consequence, there is an increasing demand for means of fine-grained traffic analysis to help gain better understanding on the activities of a computer or a small subnet.

Motivated by the needs for managing and securing end hosts and small subnets, in this paper, we propose a tool for *H*ost and netw*O*rk *S*ystem *P*rofiling and *I*nternet *T*raffic *A*na*L*ysis (HOSPITAL). HOSPITAL is designed explicitly to fulfill the needs of traffic analysis at the level of end hosts or small subnets. The system is characterized with high *deployability* and *flexibility*. The system is a self-contained tool, which can be readily placed at either an end host or at the border router of a small subnet providing concise and more meaningful network traffic summary statistics. These statistics are readily available for demonstration in a web interface and can be configured (e.g., adding or removing different modules and changing the display layout) according to specific requests from users. The operators can also retrieve these statistics remotely for diagnosing problems happening at end hosts.

HOSPITAL consists of four components, traffic collector, parser, analysis engine and a GUI. The traffic collector gathers raw network traffic data from/towards the end host or passing through the border router of a subnet. The parser extracts packet header information from the raw data and translates it into a unified flow format. The analysis engine integrates a variety of state-of-the-art traffic analysis techniques, such as RU analysis [3], traffic graph analysis [4] and block analysis [5], etc., to distill concise and meaningful summary statistics to help users gain better understanding on the ongoing traffic activities. To achieve better efficiency and ease of maintenance, the analysis engine adopts a *modularization architecture*, in which these modules are organized in a hierarchy, where high-level modules can utilize low-level modules to generate intermediate results without computing statistics from the raw data. The fourth component is a GUI, which provides summarization tools via a web-based interface that enables users to configure the HOSPITAL to their specific needs and provides users and operators with visualized analysis results.

We implement the system and deploy it to a lab subnet with

978-1-4244-8864-3/10/$26.00 ©2010 IEEE

various types of machines. Experimental results show that the system is capable of providing a variety of useful statistics, which help us gain insight on the activities associated with the end host/subnet, and detect a number of suspicious behaviors.

Related Work. With the same purpose to protect networks and end hosts, classical intrusion detection/prevention systems employ either machine learning/data mining techniques [1] or expert rules [2] to detect network and system anomalies. Unlike our system, which provides detailed information on the types of activities and their characteristics, these existing works are purpose-specific, and designed primarily to detect anomalies. Many recent works focus on profiling network/host activities. For example, Xu et. al. [6] designed and implemented a real-time Internet backbone traffic profiling system. Similarly, BLINC [7] aims at classifying traffic flows by exploring the interaction patterns of hosts and port numbers. Our work, on the other hand, provides a much finer-grained traffic analysis tool at the level of end hosts or small subnets. A number of these discussed profiling methods provide an "infrastructure-based" monitoring which relies on the collaboration of different entities, compared to HOSPITAL, which can be tuned to work as an independent self-contained system or collaboratively.

The rest of the paper is organized as follows. We discuss the challenges and design principles of HOSPITAL in Section II. The details of the design of HOSPITAL are presented in Section III. In Section IV, we demonstrate how to use HOSPITAL to understand host activities and detect anomalies and Section VI concludes the paper.

II. Design Challenges and Principles

In this section, we first discuss the challenges that users and network operators encounter when attempting to gain an insight into "what is going on" within their networks or hosts We also show how these challenges are further complicated by the intricacy the Internet has experienced during the past decade. Next, we present the design principles of HOSPITAL which not only directly address these challenges, but also provide rich analysis via a configuration utility and with little overhead making the deployment of HOSPITAL a simple task.

A. Challenges

Levels of granularity: Users and network operators are generally faced with the question of how their network resources are being utilized. While a user may be interested in the activities within his/her own machine or few machines connected to his/her home network router, an operator managing a subnet (e.g. servers within an engineering department), has the focus of all incoming traffic into these machines. Yet, security administrators managing enterprise networks have the larger scope of the overall activities passing through some vantage points or a border router. Whatever is the network segment under study (host, subnet, or whole network), users and operators may be interested in different levels of analysis. For example, while an operator of a subnet may be interested in a more fine-grained analysis, such as the activities of a specific IP address block that has been reported to send scanning traffic or an email/SMTP machine receiving large amount of spam messages, a user, on the other hand, may be interested in more aggregated information, such as major activities on personal machine or home network answering questions such as how much of gaming traffic is captured at the home router or why there is incoming p2p traffic! All these and similar questions need to be answered via easy-to-use tools that provide different granularity levels in terms of observation window times, segments of the network or even a specific IP or port activity.

Lack of customization tools: Most of the existing tools come with an overwhelming functionality for which the user may not be interested in. Not only this tends to effect deployability, but also makes the customization of these tools a tedious task for both users and developers. Users may be interested in a partial subset of functionality for which existing tools usually do not offer an easy-to-use method allowing them to choose the type and level of analysis. Moreover, interested developers may want to take these tools one step further by utilizing the source code to build more functionality that services their specific needs, which may incur a learning curve on developers to understand how the code is written. In order for a tool to be of use to developers, the design should make the incorporation of a new functionality as easy as a plug-and-play.

Deployability issue: The overhead incurred by deploying a tool plays a major factor in its deployability. Even though these existing run in the background, they fork several processes and consume a large share of CPU and memory. Moreover, in order to deploy these tools, other heavy-duty third-party software programs may need to be installed. Another limitation of existing tools is the fact that they are platform-dependent and an exerted effort is required to port them to other platforms.

Presentation and configuration issues: The majority of existing tools present analysis in the form of logs and requires experienced support engineers to sift through these logs to capture important observations and analysis. Therefore, a user finds such tools complicated to utilize. Moreover, system administrators may not be interested in spending hours staring at logs and rather prefer a summarization of findings which can be presented in a graph or a plot via a web-browser GUI for which the majority of existing tools do not offer. Moreover, most of the analysis of these tools is post-mortem. However, there are times when a user or an operator requires on-the-fly real-time analysis. Therefore, to be of use to users, a tool should incorporate some visualization capability and different modes of operation with an easy configuration mechanism.

B. Design Principles

1) Multi-level analysis: To address the levels of granularity challenge, HOSPITAL is designed to provide analysis at different levels of granularity in terms of network size (a host or a /24 vs. /16 subnet). The tool can be installed on a single machine, a switch of a subnet or a border router with no additional customization cost. Therefore HOSPITAL has the capability of providing global- as well as local-level analysis. Moreover, HOSPITAL can be configured to provide analysis for different time windows (per hour, day, week or month),

as well as different types of analysis for an application, an IP address or a block, or a specific port. Finally, the level of analysis can also be fine-grained such as an application-specific (p2p or HTTP) or coarser-grained (e.g., the total number of flows per every 5 minute through the vantage point).

2) Modular design: HOSPITAL comes with a configuration utility which addresses the lack of customization challenge by allowing users and developers to setup and customize HOSPITAL to their needs. For users, the utility provides easy-to-use XML configuration module that allows them to choose the types of analyzers to be executed and the mode of operation. For developers, the utility provides an interface module that allows them to write their own modules and easily integrate them with the existing analysis engine in a plug-and-play fashion. Moreover, HOSPITAL does not implement each analyzer by directly utilizing the raw traffic data. Instead, the analyzers follow a dependency tree from which one higher-level analyzer utilizes the statistics or outcome data obtained from another lower-level analyzer(s). For example, the RU analyzer utilizes information from the flow analyzer. For users, this tends to lower the overhead of HOSPITAL especially when it is utilized in real-time mode. For developers, this characteristic allows for developing new analyzers with minimal integration by simply utilizing the interface module which defines the dependency tree of the analyzers.

3) Lightweight design: HOSPITAL addresses the challenge of deployability issues by incurring little (to no) overhead in terms of deployment and operation. The components of HOSPITAL are either lightweight components, such as SQLite (back-end DBMS) or out-of-the-box tools such as tcpdump or wireshark. This self-contained lightweight design allows for the deployment of HOSPITAL on any platform.

4) Configurability: One of the key features of HOSPITAL is its presentation and configuration capability. First, the GUI is XML-based which allows for simple interpreters to be written easily into the targeted platform. Not only developers can customize the GUI, but users can also decide what to be presented and displayed on the HOSPITAL GUI. Second, the GUI is rich with visualization tools. Instead of going through thousands of lines of logs (as is the case in some tools), a user or a network operator can obtain analysis results in a compact format via the GUI. Finally, HOSPITAL can be configured in one of two modes: real-time or off-line, which allows for both on-the-fly and post-mortem analysis.

III. HOSPITAL Design

In this section, we present the architecture of HOSPITAL and highlight the analysis and profiling utility of hospital. Fig. 1 illustrates the architecture of HOSPITAL, which consists of four major components: a data collector, a parser, an analysis engine and a GUI.

A. Data Collector

The task of the data collector is to capture traffic from/towards a host or within a subnet. The collector could be placed on a host or connected to a switch in a subnet. For better deployability, the collector utilizes out-of-the-box tools, such as tcpdump and wireshark to collect packets (or flows, depending on the tools) and then pipes the data to the parser.

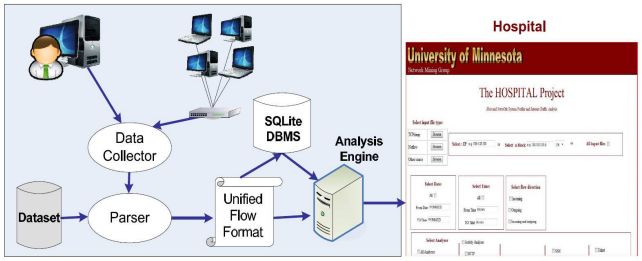


Fig. 1. Architecture of HOSPITAL and the user interface

B. Parser

Using the raw data, the parser processes the data into a common format: the unified flow format (UFF). In addition to parsing data from the data collector, the parser also supports datasets provided from readily available sources, such as Cisco NetFlow data or other data provided by ISPs or enterprises. Once parsed, UFF records are either loaded into a light-weight DBMS (SQLite) for off-line (post-mortem) analysis or piped directly to the analysis engine for real-time analysis. We note that the collector and the analysis engine can be installed in different machines, in this case, the communication between the collector and the analysis engine utilizes sockets.

C. Analysis Engine

The analysis engine is a set of analyzers each of which is referred to as an *analyzer* or a *profiler* [1], which utilizes the UFF records to provide the analysis and profiling utility. The modular design of the analysis engine eases both development and operation. New analysis method can be incorporated in a fast way and users can choose appropriate analyzers to fit their specific operation needs.

Though analyzers provide different types of measurements for the collected traffic, they are not designed in a "flat" way. Instead, these analyzers form a hierarchy. A few basic analyzers generate statistics directly from UFF records; while some analyzers are built on top of other analyzers. For example, the flow analyzer directly processes UFF records and provides flow level statistics as output. The session analyzer directly takes these flow level measurements from the flow analyzer as input and produces session level statistics. This hierarchical design reduces computation costs and overlapping processing of data. Below, we provide details on different analyzers. Due to space limit, we only provide a subset of these analyzers which will be used for experiments in Section IV.

Flow Analyzer. The flow analyzer is the most basic analyzer, which generates basic flow information for both incoming and outgoing flows. More specifically, it generates statistics for incoming matched flows, outgoing matched flows, incoming unmatched flows and outgoing unmatched flows [2]. Statistics include the number of flows, packets, bytes, port numbers, IP

[1] We use analyzer as a generic term for analyzer and profiler.

[2] A flow is considered as matched if the same flow 5-tuple is observed on the opposite direction within a time interval T, say, 30 minutes.

addresses, blocks and ASes. This rather simple analyzer captures traffic patterns that may provide insight on a dominance of some ports or a remote host (based on the number of flows). For example, a dominant port along with a dominant remote host towards a local block may imply the presence of certain incoming scanning activity.

Session Analyzer. Session analyzer is another basic analyzer that provides session statistics, such as number of sessions initiated by a local host for a given period of time, number of sessions initiated by the remote host, number of flows per session, duration of session, session with maximum/minimum flows or duration, etc. Session information can be utilized to obtain some observations on the types of sessions a given host is involved in. For example, presence of sessions, such as HTTP 3-way handshake, that do not terminate may be a sign of suspicious activity (i.e. SYN flood attack).

RU Analyzer. This analyzer utilizes the statistics collected by the flow analyzer to compute the *relative uncertainty* (RU) vector of the local host or block based on parameters specified by the user. These parameters include the number of flows sufficient to calculate RU values [3] (e.g. 100 TCP flows). RU vector contains RU values for source/destination IPs, source/destination ports, number of flows, packets or bytes, which measures the dominance (or lack thereof) of an IP address, a port, or traffic volume. The RU analyzer can be used for different purposes, such as finding dominant IPs or ports. RU can also be used to distinguish between dynamic and static remote addresses (see [8] for details).

Activity Patterns Profiler. The activity patterns profiler captures the significant activity patterns of a subnet. The activity patterns profiler utilizes the pLSA method as explained in [9]. In pLSA, the host-port association is represented in a matrix A, the entries of which $A(h, p)$ represents the joint probability that port p is associated with host h in the overall traffic. The joint probability is defined by the mixture $A(h, p) := Pr(h, p) = \sum_{c \in C} Pr(p|c)Pr(c|h)Pr(h)$, in which $C = \{c_1, \ldots, c_K\}$ represents the K latent applications (or "activity patterns") within the subnet. The method generates the distinct activity patterns with interpretable labels.

All these aforementioned analyzers can be configured to only focus on specific applications, such as DNS traffic or ICMP traffic. We can also zoom in into a limited number of hosts or blocks of interest. For example, tracking the activities of certain "suspicious" or blacklisted hosts.

D. GUI and Interface

Due to the fact that analyzers provide different types of traffic statistics, a major effort is imposed on delivering such information to users in a more clear and concise way. HOSPITAL comes with an XML-based GUI that can easily be interpreted into a platform. Moreover, it can be easily modified and customized by the user. The power in the GUI lies also in its visualization capabilities of providing visual representation of statistics, communities, and interactions.

[3]RU for m observations is calculated as $\frac{-\sum i=1^m p_i log_2 p_i}{log_2 m}$ in which p_i is the percentage (or frequency) of observing i in m.

IV. Experiment Results

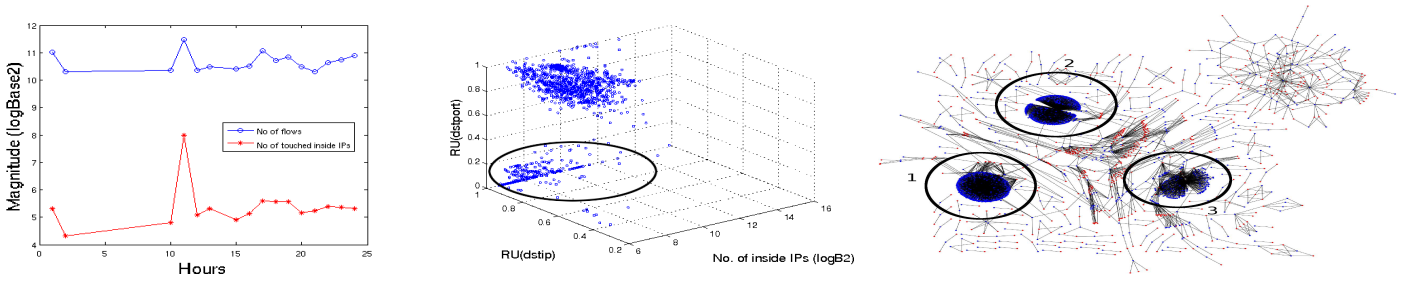
In this section, we demonstrate the analysis and profiling utility of HOSPITAL by applying some of its analyzers on data collected at our network.

Experiment Setup: We installed HOSPITAL on a switch of our lab /24 subnet, which contains 30 active hosts. There are 10 Ubuntu Linux clients, 8 Sun Solaris machines, 4 MS Windows 7 desktops, 4 MS windows XP desktops, 2 Windows Vista desktops and 2 MAC OS machines. Users were given the option to install HOSPITAL on their own machines and the mode of operation on these machines varied between off-line and real-time modes. Users used these machines as usual and from our experience of using the machines for a week, we noticed that there was no complaint regarding the overhead associated with running HOSPITAL on these machines[4]. Fig. 2(a) shows plot generated by the HOSPITAL's web-based GUI which depicts the results obtained from applying the flow analyzer to the collected traffic on the /24 block in terms of the No. of flows and No. of inside IP addresses receiving traffic from the outside. The x-axis represents the time (in hours) and the y-axis represents the magnitude (in log base 2). As shown, we observe a spike during the 11^{th} hour. Upon investigation of the data towards the inside block, we find a remote IP address sending traffic to every IP address within the inside block directed at the same *UDP port 139*. The IP address is reported on blacklists [10] and extensive research on the port shows that even though *UDP port 139* is a NetBIOS port utilized for Windows file and printer sharing, it is reported that it is usually the first port hackers target in port scan attacks.

To illustrate the effect of scanning within our collected data, we configure HOSPITAL to filter all remote IP addresses that touches at least 100 IP addresses from a /16 block. For each remote IP address, using the No. of flows, we compute $RU(dstip)$ and $RU(dstprt)$. A remote IP address sending traffic using some dominant dstprt ($RU(dstprt) \approx 0$) could be accessing inside service (p2p or web server). However, if the remote IP address sends equally the same number of flows to all inside IP addresses ($RU(dstip) \approx 1$), the remote IP address is most probably a scanner. In Fig. 2(b), HOSPITAL GUI plots the results of the RU analyzer in which the x-axis represents the number of local IP addresses touched by the remote IP address (in log base 2), the y-axis represents $RU(dstip)$ and the z-axis represents $RU(dstprt)$. We notice that the IP addresses inside the oval have large $RU(dstip)$ and small $RU(dstprt)$. Upon validation of a sample of IP addresses, we discover that they are blacklisted as scanners.

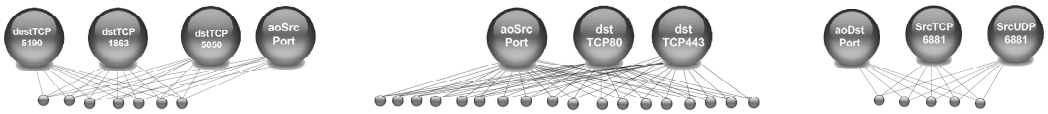
To demonstrate the multi-level design property of HOSPITAL, we install the tool on the border router of our campus network and collect traffic for a whole day. We show the utility of the application profiler by configuring the analyzer to focus on DNS traffic *UDP/TCP port 53*. Fig. 2(c), shows a graph generated by the GUI representing the interaction patterns between unsuccessful DNS queries and the IP addresses who initiate these queries in a whole-day traffic data. By extracting community structures from the graph, the analyzer detects

[4]More rigorous evaluation of the overhead is one of our future works.



(a) Flow Analyzer (b) RU Analyzer (c) DNS Failure

Fig. 2. Experiment results for three analyzers



(a) IM (b) HTTP (c) P2P

Fig. 3. Experiment results: activity patterns profiler

correlated DNS query failures caused by potentially malicious network activities. We mark 3 community structures detected by the analyzer. Communities 1 and 2 correspond to well-known domain-flux bots, Conficker A and Conficker B [11], [12]. Community 3 is caused by a host queries for many non-existing email servers, possibly related to spamming activities.

We later configure the tool at the border router to capture traffic corresponding to a /24 residential hall block. Fig . 3 shows the results of applying the activity patterns profiler to the traffic of the selected residential hall /24 block. The diagram shows the significant activity patterns [9] in which the lower row circles represent the hosts (h), the upper row circles represent the ports (p) and an edge between a host and a port shows strong utilization of p by h. We find that there are mainly three activity patterns within the selected /24 block. The first activity of this block represents Instant Messaging (IM) using AOL *dst port 5190*, MSN Messenger *dst port 1863* and Yahoo Messenger *dst port 5050*. The second activity pattern represents an HTTP web server using source *port 80* and secure HTTP (HTTPs) web server represented using source *port 443* along with *aoDstPort*. Common to residential and dynamic subnets, p2p and file sharing service are highly popular on such subnets which are depicted by activity 2 with BitTorrent *src UDP/TCP 6881*, and *aoDstPort*.

The results obtained from utilizing a few analyzers on a sample data shows the summarization tool provided to users and network administrators. While HOSPITAL may not fix any issues, it can, through its analysis, summarization and profiling capabilities provide users and operators concise as well as detailed information on possible underlying causes of a malfunction. HOSPITAL can also be used in conjunction with other tools, such as IDS, anti-virus programs, spy-ware detectors and removers along with other security tools.

V. Conclusions

In this paper, we proposed HOSPITAL, a lightweight system for multilevel profiling and analyzing network traffic at the level of end hosts and subnets. HOSPITAL consists of four components. The data collector captures traffic from a host or subnet, then passes it to the parser to convert the data into a common UFF format. The analysis engine integrates a set of analyzers to generate analysis results from the collected data. GUI component provides user friendly interface to demonstrate the analysis results. We implemented HOSPITAL in a real network and demonstrated that we could gain some insight on what is going on in within hosts and subnets. In future work, we plan to conduct evaluation on the scalability of the system and the communication overhead when HOSPITAL is deployed in a distributed environment.

VI. Acknwledgement

The work is supported in part by the National Science Foundation grants CNS-0626808, CNS-0626812, CNS-0905037 and the DTRA grant HDTRA1-09-1-0050.

References

[1] MINDS. Minnesota Intrusion Detection System. http://www.cs.umn.edu/research/minds/.
[2] SNORT. http://www.snort.org/.
[3] K. Xu, Z. Zhang and S. Bhattacharyya. Profiling Internet backbone traffic: behavior models and applications. In *Proc. of ACM SIGCOMM*, August 2005.
[4] Y. Jin, E. Sharafuddin, and Z-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *Proc. of SIGMETRICS '09*, pages 49–60, 2009.
[5] E. Sharafuddin, Y. Jin, N. Jiang, and Z.-L. Zhang. Know your enemy, know yourself: Block-level network behavior profiling and tracking. In *To appear in Proc. of GLOBECOM*, 2010.
[6] K. Xu, F. Wang, S. Bhattacharyya, and Z. Zhang. A real-time network traffic profiling system. In *DSN*, pages 595–605. IEEE Computer Society, 2007.
[7] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *In Proceedings of ACM SIGCOMM*, pages 229–240, 2005.
[8] Y. Jin and E. Sharafuddin and Z.-L. Zhang. Identifying Dynamic IP Address Blocks Serendipitously through Background Scanning Traffic. In *Proc. of ACM CoNext'07*, 2007.
[9] E. Sharafuddin, Y. Jin, N. Jiang, and Z. Zhang. Sifting through network data to cull activity patterns with heaps. In *Proc. of ICDCS*, 2010.
[10] MX Toolbox Blacklists. http://www.mxtoolbox.com/blacklists.aspx.
[11] P. Porras, H. Saidi, and V. Yegneswaran. An analysis of conficker's logic and rendezvous points. http://mtc.sri.com/Conficker/.
[12] N. Jiang, J. Cao, Y. Jin, Z.-L. Zhang, and L. Li. Identifying suspicious activities through dns failure graph analysis. Technical report, University of Minnesota, 2010.