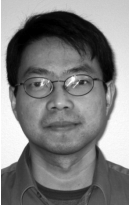


JIAN ZHANG, PHILLIP PORRAS, AND
JOHANNES ULLRICH

highly predictive blacklisting



Jian Zhang is an assistant professor in the department of computer science at Louisiana State University. His research interest is in developing new machine-learning methods for improving cybersecurity.

zhang@csc.lsu.edu



Phillip Porras is a Program Director of systems security research in the Computer Science Laboratory at SRI International. His research interests include malware and intrusion detection, high-assurance computing, network security, and privacy-preserving collaborative systems.

porras@csl.sri.com



As Chief Research Officer for the SANS Institute, Johannes Ullrich is currently responsible for the SANS Internet Storm Center (ISC) and the GIAC Gold program. He founded the widely recognized DShield.org in 2000, and in 2004 *Network World* named him one of the 50 most powerful people in the networking industry.

jullrich@sans.org

WE INTRODUCE THE HIGHLY PREDICTIVE Blacklist (HPB) service, which is now integrated into the DShield.org portal [1]. The HPB service employs a radically different approach to blacklist formulation than that of contemporary blacklist formulation strategies. At the core of the system is a ranking scheme that measures how closely related an attack source is to a blacklist consumer, based on both the attacker's history and the most recent firewall log production patterns of the consumer. Our objective is to construct a customized blacklist per repository contributor that reflects the most probable set of addresses that may attack the contributor in the near future. We view this service as a first experimental step toward a new direction in high-quality blacklist generation.

For nearly as long as we have been detecting malicious activity in networks, we have been compiling and sharing blacklists to identify and filter the most prolific perpetrators. Source blacklists are a fundamental notion in collaborative network protection. Many blacklists focus on a variety of illicit activity. Network and email address blacklists have been around since the earliest days of the Internet. However, as the population size and personal integrity of Internet users have continued to grow in inverse directions, so too have grown the popularity and diversity of blacklisting as a strategy for self-protection. Recent examples include source blacklists to help networks detect and block the most prolific port scanners and attack sources, SPAM producers, and phishing sites, to name a few [2, 3, 8].

Today, sites such as DShield.org not only compile global worst offender lists (GWOLs) of the most prolific attack sources, but they regularly post firewall-parsable filters of these lists to help the Internet community fight back [8]. DShield represents a centralized approach to blacklist formulation, with more than 1700 contributors providing a daily perspective on the malicious background radiation that plagues the Internet [6, 9]. DShield's published GWOL captures a snapshot of those class C subnets whose addresses have been logged by the greatest number of contributors.

Another common approach to blacklisting is for a local network to create its own local worst offender list (LWOL) of those sites that have attacked it the most. LWOLs have the property of capturing repeat offenders that are indeed more likely to return to the local site in the future. However, the LWOL-based blacklisting strategy is an inherently reactive technique, which asserts filters against network addresses that have been seen to flood, probe, or conduct intrusion attempts against local network assets. LWOLs have the property of capturing repeat offenders that are indeed more likely to return to the site in the future, thus effectively reducing unwanted traffic. However, by definition an LWOL cannot include an address until that address has demonstrated significant hostility or has saturated the local network with unwanted traffic.

The GWOL-based blacklisting strategy addresses the inherent reactivity of LWOL strategies by extending the observation pool of malicious source detectors. A GWOL attempts to capture and share a consensus picture from many collaborating sites of the worst sources of unwanted network traffic. Unlike LWOLs, GWOLs have the potential to inform a local network of highly prolific attackers, even when those attackers have not yet been seen by the network. Unfortunately, the GWOL strategy also has measurable limitations. For example, GWOLs often provide subscribers with a list of addresses that may simply never be encountered at their local sites. Malware also provides a significant challenge to GWOLs. A widely propagating indiscriminate worm may produce a large set of prolific sources—but what impact do a few hundred entries make where there are tens of thousands of nodes that would qualify as prolific? Alternatively, a botnet may scan a large address range cooperatively, where no single bot instance stands out as the most prolific.

The HPB Blacklisting System

A high-quality blacklist that fortifies network firewalls should achieve high hit rates, should incorporate addresses in a timely fashion, and should proactively include addresses even when they have not previously been encountered by the blacklist consumer's network. Toward this goal, we present a new blacklist-generation strategy, which we refer to as highly predictive blacklisting.

To formulate an HPB for a given DShield contributor, we assign a rank score to each attack source address within the repository. The rank score reflects the degree to which that attacker has been observed by other contributors who share a degree of overlap with the target HPB owner. The ranking score is derived not by considering how many contributors the source has attacked in the past (which is the case in formulating the worst offender list), but, rather, by considering which contributors it has attacked. The HPB framework also employs another technique to estimate a source's attack probability even when it has been observed by only a few contributors. This technique models the contributors and their correlation relationship as a graph. The initial attack probability derived from the evidence (the few attacks reported) gets propagated within this graph, and the ranking score is then inferred using the propagated probability. Our methodology employs a random walk procedure similar to the Google PageRank link analysis algorithm [11].

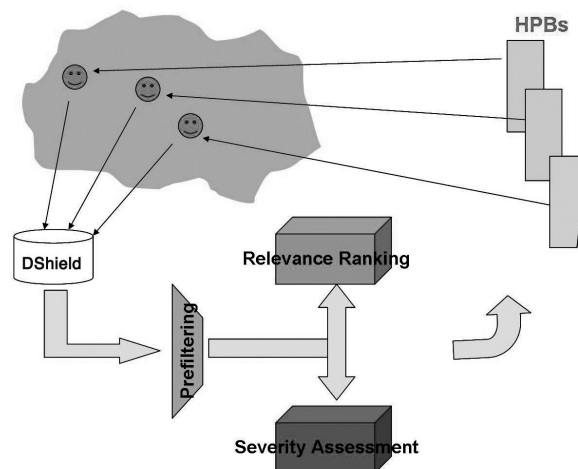


FIGURE 1: BLACKLISTING SYSTEM ARCHITECTURE

As illustrated in Figure 1 (preceding page), our system constructs blacklists in three stages. First, security logs supplied by DShield undergo preprocessing to filter false-positive or likely innocuous alerts (noise) within the DShield data repository. The filtered data is fed into two parallel analysis engines. The first engine ranks all attack sources, per contributor, according to their relevance to that contributor. The second engine scores the sources by using a severity assessment that measures their maliciousness. The resulting relevance rankings and severity scores are then combined to generate a final blacklist for each contributor.

We consider three noise-filtering techniques. First, we remove DShield logs produced from attack sources from invalid or unassigned IP address space. We employ the bogon list created by the Cymru team to identify addresses that are reserved, not yet allocated, or delegated by the Internet Assigned Number Authority [7]. Second, we prefilter network addresses from Internet measurement services, Web crawlers, and common software updating services. We have developed a whitelist of such sources that often generate alarms in DShield contributor logs. Finally, we apply heuristics to avoid common false-positives that arise from timed-out network services. Specifically, we exclude logs produced from source ports TCP 53 (DNS), 25 (SMTP), 80 (HTTP), and 443 (often used for secure Web, IMAP, and VPN) and from destination ports TCP 53 (DNS) and 25 (SMTP). In practice, the combination of these prefiltering steps provides approximately a 10% reduction in the DShield input stream prior to delivery into the blacklist-generation system.

The two analysis engines are the core of our blacklisting system. In relevance analysis, we produce an “importance” measurement for an attacker with respect to a particular blacklist consumer. With this measurement, we try to capture the likelihood that the attacker may come to the blacklist consumer in the near future.

In the system, the blacklist consumers are the contributors that supply security logs to a log-sharing repository such as DShield. Consider a collection of security logs displayed in a tabular form (Table 1). We use the rows of the table to represent attack sources (*attackers*) and the columns to represent contributors (*victims*). An asterisk (*) in the table cell indicates that the corresponding source has reportedly attacked the corresponding contributor.

	v_1	v_2	v_3	v_4	v_5
s_1	*	*			
s_2	*	*			
s_3	*		*		
s_4		*	*		
s_5		*			
s_6				*	*
s_7			*		
s_8			*	*	

TABLE 1: SAMPLE ATTACK TABLE

Suppose we would like to calculate the relevance of the attack sources for contributor v_1 based on these attack patterns. From the attack table we see that contributors v_1 and v_2 share multiple common attackers; v_1 also shares one common attack source (s_3) with v_3 , but not with the other contributors. Given this observation, between sources s_5 and s_6 , we would say that s_5 has more relevance to v_1 than s_6 , because s_5 has reportedly attacked v_2 , which has recently experienced multiple attack source overlaps with v_1 , whereas the victims of s_6 's attacks share no overlap with v_1 or v_2 . Note that this relevance measure is quite different from the measures based on how prolific the attack source has been. The latter would favor s_6 over s_5 , as s_6 has attacked more victims than s_5 . In this sense, *which* contributors a source has attacked is of greater significance to our scheme than how many victims it has attacked. Similarly, between s_5 and s_7 , s_5 is more relevant, because the victim of s_5 (v_2) shares more common attacks with v_1 than the victim of s_7 (v_3). Finally, because s_4 has attacked both v_2 and v_3 , we would like to say that it is the most relevant among s_4 , s_5 , s_6 , and s_7 .

We can go a step forward from this simple relevance calculation to provide more desirable properties. For example, the set of contributors consists of only a very small set of networks in the Internet. Before an attacker saturates the Internet with malicious activity, it is often the case that only a few contributors have observed the attacker. For example, the attacker may be at an early stage in propagating attacks, or it may be a prolific scanner for networks that do not participate in the security log sharing system. Therefore, one may want to take into consideration possible future observations of the source and include these anticipated observations from the contributors into the relevance values.

This can be achieved through a relevance propagation process. We model the attack correlation relationship between contributors using a correlation graph $G = (V, E)$. The nodes in the graph are the contributors $V = \{v_1, v_2, \dots, v_n\}$. There is an edge between node v_i and v_j if v_i is correlated with v_j . The weight on the edge is determined by the strength of the correlation. If a contributor v_i observed an attacker, we say that the attacker has an initial relevance value 1 for that contributor. Following the edges that go out of the contributor, a fraction of this relevance can be distributed to the neighbors of the contributor in the graph. Each of v_i 's neighbors receives a share of relevance that is proportional to the weight on the edge that connects the neighbor to v_i . Suppose v_j is one of the neighbors. A fraction of the relevance received by v_j is then further distributed, in similar fashion, to its neighbors. The propagation of relevance continues until the relevance values for each contributor reach a stable state.

Figure 2 gives an example of this propagation feature. The correlation graph of Figure 2 consists of four contributors numbered 1, 2, 3, and 4. Contributor 1 reported an attack from source s . Our goal is to evaluate how relevant this attacker is to contributor 4. Although, at this time, contributors 2 and 3 have not observed s yet, there may be possible future attacks from s . In anticipation of this, when evaluating s 's relevance with respect to contributor 4, contributors 2 and 3 pass to contributor 4 their relevance values after multiplying them with the weights on their edges, respectively. The attacker's relevance value for contributor 4 then is $0.5 * 0.2 + 0.3 * 0.2 = 0.16$. Note that had s actually attacked contributors 2 and 3, the contributors would have passed the relevance value 1 (after multiplying them with the weights on the edges) to contributor 4.

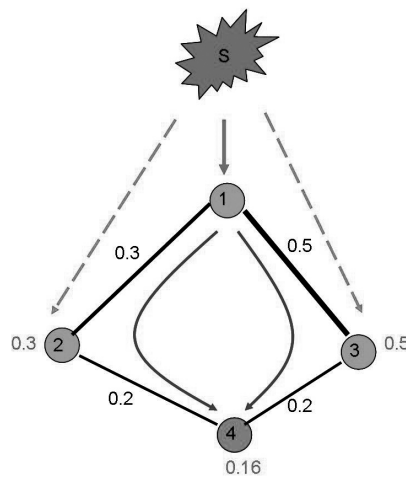


FIGURE 2: RELEVANCE EVALUATION CONSIDERS POSSIBLE FUTURE ATTACKS

Let W be the adjacency matrix of the correlation graph, where the entry $W_{(i,j)}$ in this matrix is the weight of the edge between nodes v_j and v_i . For a source s , we use a vector b_s to indicate the set of contributors that have reported an attack from s . ($b_s = \{b_1^s, b_2^s, \dots, b_n^s\}$ such that $b_i^s = 1$ if $v_i \in T(s)$ and $b_i^s = 0$ otherwise.) We also associate with each source s a relevance vector $r^s = \{r_1^s, r_2^s, \dots, r_n^s\}$ such that r_v^s is the relevance value of attacker s with respect to contributor v . After the propagation process, the relevance vector would become

$$r^s = \sum_{i=1}^n (aW)^i \cdot b^s$$

We observe that $b_s + r^s$ is the solution for x in the following system of linear equations:

$$\mathbf{x} = \mathbf{b}^s + \mathbf{a}\mathbf{W}\cdot\mathbf{x}$$

The linear system described by Equation 2 is exactly the system used by Google's PageRank [1]. PageRank analyzes the link structures of Web pages to determine the relevance of each Web page with respect to a keyword query. Similarly, our relevance values reflect the structure of the correlation graph that captures intrinsic relationships among the contributors.

The second analysis engine in our system is the severity assessment engine. It measures the degree to which each attack source exhibits known patterns of malicious behavior. We focus on behavior patterns of an attacker who conducts an IP sweep to small sets of ports that are known to be associated with malware propagation or backdoor access as documented by Yegneswaran et al. [9], as well as our own most recent experiences (within the past year) of more than 20,000 live malware infections observed within our honeynet [10]. Other potential malware behavior patterns may be applied (e.g., the scan-oriented malicious address detection schemes outlined in the context of dynamic signature generation [5] and malicious port scan analysis [4]). Regardless of the malware behavior model used, the design and integration of other severity metrics into the final blacklist-generation process can be carried out in a similar fashion.

Besides ports that are commonly associated with malware activities, we also consider the set of unique target IP addresses to which attacker s is connected. A large unique IP count represents confirmed IP sweep behavior, which can be strongly associated with our malware behavior model. Third, we compute an optional tertiary behavior metric that captures the ratio of national to international addresses that are targeted by attacker s , $IR(s)$. Within the DShield repository we find many cases of sources (such as from China, Russia, and the Czech Republic) that exclusively target international victims.

Once each attacker is processed by the two analysis engines, we have both their relevance rankings and their severity scores. We can combine them to generate a final blacklist for each contributor. We would like to include the attackers that have strong relevance and also show malicious behavior patterns. To generate a final list, we use the attacker's relevance ranking to compile a candidate list of double the intended size and then use severity scores of the attackers to adjust their ranking on the candidate list. The adjustment promotes the rank of an attacker if the severity assessment indicates that it is very malicious. The final blacklist is formulated by picking the top-ranked attackers.

Experiment Results

To evaluate our HPB blacklist formulation system we performed a battery of experiments using the DShield.org security firewall and IDS log repository. We examined a collection of more than 720 million log entries produced by DShield contributors from October to November 2007.

To assess the performance of the HPB system, we compare its performance relative to the standard DShield-produced GWOL [8]. In addition, we compare our HPB performance to that of LWOLs, which we compute individually for all contributors in our comparison set. We generate GWOL, LWOL, and HPBs using data for a certain time period and then test the blacklists on data from the time window following this period. Performance is determined by how many entries on a list are encountered in the testing window. For the purpose of our comparative assessment, we fixed the length of all three competing blacklists to exactly 1000 entries. Additional experiments show that the results are consistent over time, across various list lengths and testing windows.

Table 2 (next page) shows the total number of hits summed over the contributors for HPB, GWOL, and LWOL, respectively. It also shows the ratio of HPB hits over that of GWOL and LWOL. Overall, HPBs predict 20%–30% more hits than LWOL and GWOL.

Window	GWOL total hits	LWOL total hits	HPB total hits	HPB/GWOL	HPB/LWOL
1	81,937	85,141	112,009	1.36701	1.31557
2	83,899	74,206	115,296	1.37422	1.55373
3	87,098	96,411	122,256	1.40366	1.26807
4	80,849	75,127	115,715	1.43125	1.54026
5	87,271	88,661	118,078	1.353	1.33179
6	93,488	73,879	122,041	1.30542	1.6519
7	100,209	105,374	133,421	1.33143	1.26617
8	96,541	91,289	126,436	1.30966	1.38501
9	94,441	107,717	128,297	1.35849	1.19106
10	96,702	94,813	128,753	1.33144	1.35797
11	97,229	108,137	131,777	1.35533	1.21861
Average	90,879 6851	90,978 13002	123,098 7193	1.36 0.04	1.37 0.15

TABLE 2: HIT NUMBER COMPARISON AMONG HPB, LWOL, AND GWOL

The results in Table 2 show the HPB hit improvement over various time windows. We now investigate the distribution of the HPB's hit improvement across contributors in one time window. In Figures 3 and 4 we plot relative hit count improvement (RI), which is the ratio in percentage of the HPB hit count increase over the other blacklist hit count.

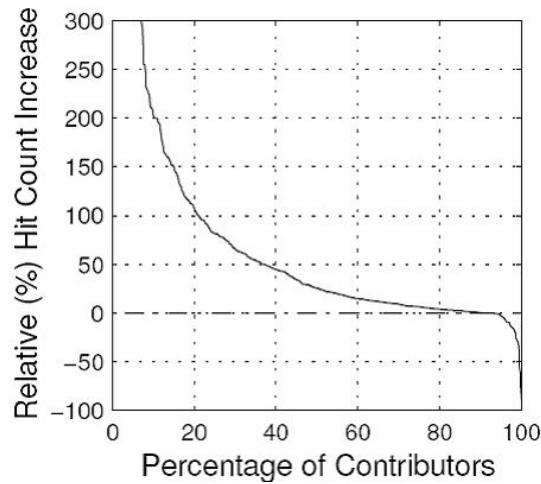


FIGURE 3: HIT COUNT COMPARISON OF HPB AND GWOL

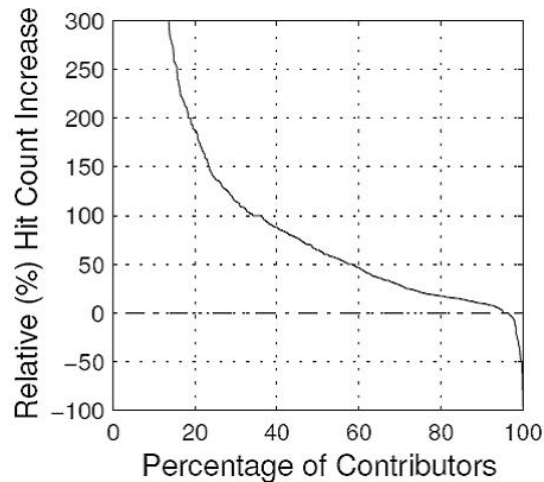


FIGURE 4: HIT COUNT COMPARISON OF HPB AND LWOL

In comparison to GWOL, there are about 20% of contributors for which the HPBs achieve an RI more than 100 (i.e., the HPB at least doubled the GWOL hit count). For about half of the contributors, the HPBs have about 25% more hits (an RI of 25). The HPBs have more hits than GWOL for almost 90% of the contributors. Only for a few contributors (about 7%) do HPBs perform worse. With LWOL, the RI values exhibit similar distributions. Note that HPBs perform worse for a small group of contributors. Further experiments show that this occurs because the HPBs' performance is not consistent for these contributors (i.e., in some time windows HPBs perform well, but in others they perform worse). We suspect that for this group of contributors the attack correlation is not stable or the attacker population is very dynamic, so it is difficult to make consistent prediction. Our experiments indicate that there is only a small group of contributors that exhibit this phenomenon. For most of the contributors, the HPBs performance is consistent.

Conclusion

We introduced a new system to generate blacklists for contributors to a large-scale security-log sharing infrastructure. The system employs a link analysis method similar to Google's PageRank for blacklist formulation. It also integrates substantive log prefiltering and a severity metric that captures the degree to which an attacker's alert patterns match those of common malware-propagation behavior. Experimenting on a large corpus of real DShield data, we demonstrate that our blacklists have higher attacker hit rates and long-term performance stability.

In April of 2007, we released a highly predictive blacklist service at DShield.org. The HPB is a free service available to DShield's log contributors, and to date the service has a pool of roughly 70 downloaders. We believe that this service offers a new argument to help motivate the field of secure collaborative data-sharing. In particular, it demonstrates that people who collaborate in blacklist formulation can share a greater understanding of attack source histories and thereby derive more informed filtering policies. As future work, we will continue to evolve the HPB blacklisting system as our experience grows through managing the blacklist service.

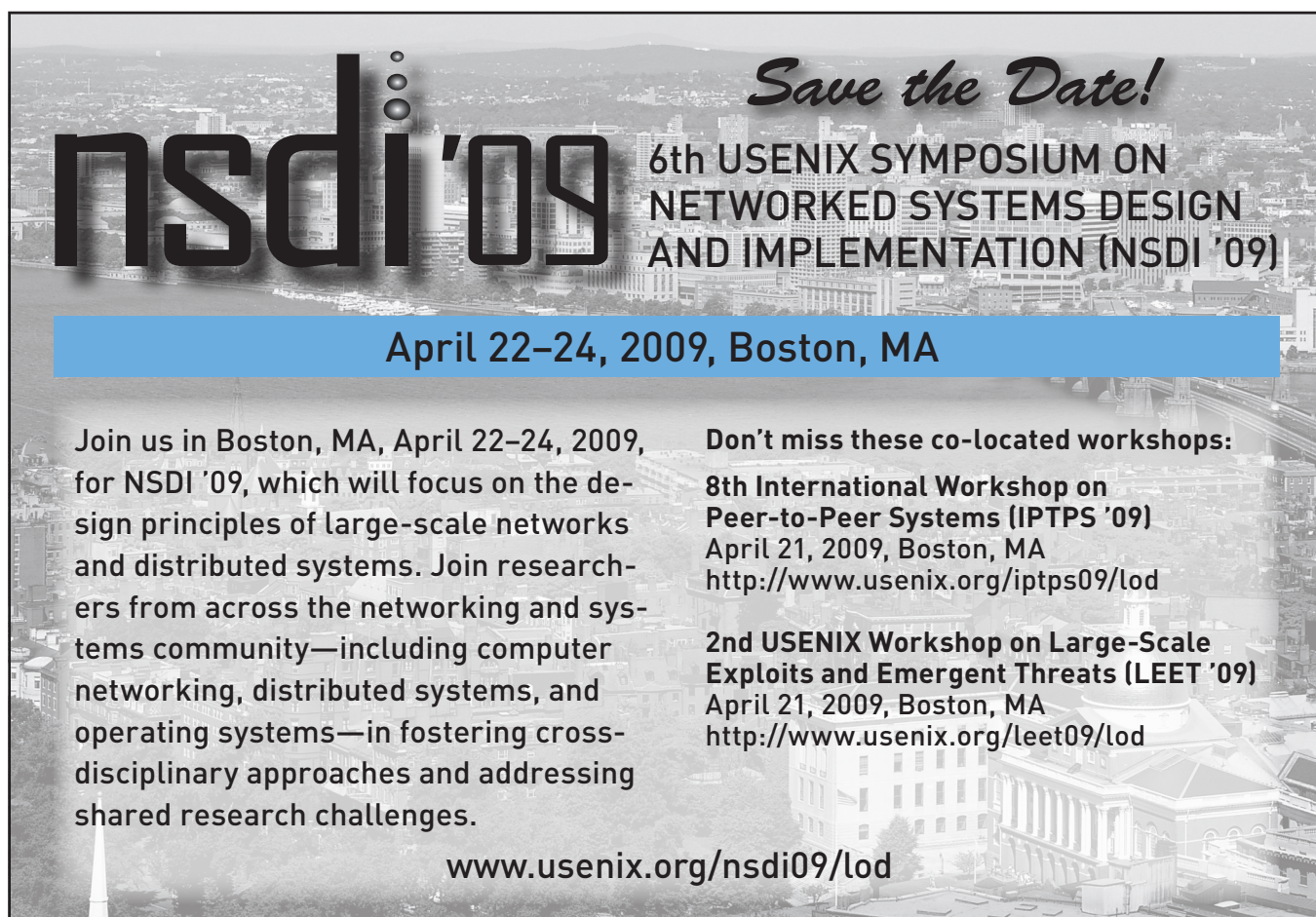
ACKNOWLEDGMENTS

This material is based upon work supported through the U.S. Army Research Office under the Cyber-TA Research Grant No. W911NF-06-1-0316.

REFERENCES

- [1] J. Zhang, P. Porras, and J. Ullrich, DSHIELD highly predictive blacklisting service: <http://www.dshield.org/hpbinfo.html>.
- [2] Google list of blacklists: <http://directory.google.com/Top/Computers/Internet/Abuse/Spam/Blacklists/>.
- [3] Google live-feed anti-phishing blacklist: <http://sb.google.com/safebrowsing/update?version=goog-black-url:1:1>.
- [4] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast Portscan Detection Using Sequential Hypothesis Testing," *IEEE Symposium on Security and Privacy 2004*, Oakland, CA, May 2004.
- [5] H.-A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," *2004 USENIX Security Symposium*, pp. 271–286.
- [6] P. Ruoming, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of Internet Background Radiation," *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2004.

- [7] R. Thomas, Bogon dotted decimal list v3.9: <http://www.cymru.com/Documents/bogon-dd.html>.
- [8] J. Ullrich, DShield global worst offender list: <https://feeds.dshield.org/block.txt>.
- [9] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet Intrusions: Global Characteristics and Prevalence," *Proceedings of ACM SIGMETRICS*, June 2003.
- [10] V. Yegneswaran, P. Porras, H. Saidi, M. Sharif, and A. Narayanan, Cyber-TA compendium honeynet page: <http://www.cyber-ta.org/Honeynet>.
- [11] 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.



nsdi'09 *Save the Date!*
6th USENIX SYMPOSIUM ON
NETWORKED SYSTEMS DESIGN
AND IMPLEMENTATION (NSDI '09)

April 22-24, 2009, Boston, MA

Join us in Boston, MA, April 22-24, 2009, for NSDI '09, which will focus on the design principles of large-scale networks and distributed systems. Join researchers from across the networking and systems community—including computer networking, distributed systems, and operating systems—in fostering cross-disciplinary approaches and addressing shared research challenges.

Don't miss these co-located workshops:

- 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)**
April 21, 2009, Boston, MA
<http://www.usenix.org/iptps09/lod>
- 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '09)**
April 21, 2009, Boston, MA
<http://www.usenix.org/leet09/lod>

www.usenix.org/nsdi09/lod