Highly Predictive Blacklisting

Jian Zhang
SRI International
Menlo Park, CA 94025

Phillip Porras
SRI International
Menlo Park, CA 94025

Johannes Ullrich
SANS Institute
Bethesda, MD 20814

Abstract

The notion of blacklisting communication sources has been a well-established defensive measure since the origins of the Internet community. In particular, the practice of compiling and sharing lists of the worst offenders of unwanted traffic is a blacklisting strategy that has remained virtually unquestioned over many years. But do the individuals who incorporate such blacklists into their perimeter defenses benefit from the blacklisting contents as much as they could from other list-generation strategies? In this paper, we will argue that there exist better alternative blacklist generation strategies that can produce higher-quality results for an individual network. In particular, we introduce a blacklisting system based on a relevance ranking scheme borrowed from the link-analysis community. The system produces customized blacklists for individuals who choose to contribute data to a centralized log-sharing infrastructure. The ranking scheme measures how closely related an attack source is to a contributor, using that attacker's history and the contributor's recent log production patterns. The blacklisting system 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. Our intent is to yield individualized blacklists that not only produce significantly higher hit rates, but that also incorporate source addresses that pose the greatest potential threat. We tested our scheme on a corpus of over 700 million log entries produced from the DShield data center and the result shows that our blacklists not only enhance hit counts but also can proactively incorporate attacker addresses in a timely fashion. An early form of our system have been fielded to DShield contributors over the last year.

1 Introduction

A network address blacklist represents a collection of source IP addresses that have been deemed undesirable, where typically these addresses have been involved in some previous illicit activities. For example, DShield (a large-scale security-log sharing system) regularly compiles and posts a firewall-parsable blacklist of the most prolific attack sources seen by its contributors [17]. With more than 1700 contributing sources providing a daily stream of 30 million security log entries, such daily blacklists provide an informative view of those class C subnets that are among the bane of the Internet with respect to unwanted traffic. We refer to the blacklists that are formulated by a large-scale alert repository and consist of the most prolific sources in the repository's collection of data as the *global worst offender list* (GWOL). Another strategy for formulating network address blacklists is for an individual network to create a local blacklist based entirely on its own history of incoming communications. Such lists are often culled from a network's private firewall log or local IDS alert store, and incorporate the most repetitive addresses that appear within the logs. We call this blacklist scheme the *local worst offender list* (LWOL) method.

The GWOL and LWOL strategies have both strengths and inherent weaknesses. For example, while GWOLs provide networks with important information about highly prolific attack sources, they also have the potential to exhaust the subscribers' firewall filter sets with addresses that will simply never be encountered. Among the sources that do target the subscriber, GWOLs may miss a significant number of attacks, in particular when the attack sources prefer to choose their targets more strategically, focusing on a few known vulnerable networks [4]. Such attackers are not necessarily very prolific and are hence elusive to GWOLs. The sources on an LWOL have repetitively sent unwanted communications to the local network and are likely to continue doing so. However, LWOLs are limited by being entirely reactive – they only capture attackers that have been pounding the local network and hence cannot provide a potential for the blacklist consumer to learn of attack sources before

these sources reach their networks.

Furthermore, both types of lists suffer from the fact that an attack source does not achieve candidacy until it has produced a sufficient mass of communications. That is, although it is desirable for firewall filters to include an attacker's address *before* it has saturated the network, neither GWOL nor LWOL offer a solution that can provide such timely filters. This is a problem particularly with GWOL. Even after an attacker has produced significant illicit traffic, it may not show up as a prolific source within the security log repository, because the data contributors of the repository are a very small set of networks on the Internet. Even repositories such as DShield that receive nearly 1 billion log entries per month represent only a small sampling of Internet activity. Significant attacker sources may elude incorporation into a blacklist until they have achieved extensive saturation across the Internet.

In summary, a high-quality blacklist that fortifies network firewalls should achieve high hit rate, should incorporate addresses in a timely fashion, and should proactively include addresses even when they have not been encountered previously by the blacklist consumer's network. Toward this goal, we present a new blacklist generation system which we refer to as the highly predictive blacklisting (HPB) system. The system incorporates 1) an automated log prefiltering phase to remove unreliable alert contents, 2) a novel relevance-based attack source ranking phase in which attack sources are prioritized on a per-contributor basis, and 3) a severity analysis phase in which attacker priorities are adjusted to favor attackers whose alerts mirror known malware propagation patterns. The system constructs final individualized blacklists for each DShield contributor by a weighted fusion of the relevance and severity scores.

HPB's underlying relevance-based ranking scheme represents a significant departure from the long-standing LWOL and GWOL strategies. Specifically, the HPB scheme examines not just *how many* targets a source address has attacked, but also *which* targets it has attacked. In the relevance-based ranking phase, each source address is ranked according to how closely related the source is to the target blacklist subscriber. This relevance measure is based on the attack source similarity patterns that are computed across all members of the DShield contributor pool (i.e., the amount of attacker overlap observed between the contributors). Using a data correlation strategy similar to hyper-text link analysis, such as Google's PageRank [2], the relationships among all the contributors are iteratively explored to compute an individual relevance value from each attacker to each contributor.

We evaluated our HPB system using more than 720 million log entries produced by DShield contributors from October to November 2007. We contrast the performance of the system with that of the corresponding GWOLs and LWOLs, using identical time windows, input data, and blacklist lengths. Our results show that for most contributors (more than 80%), our blacklist entries exhibit significantly higher hit counts over a multiday testing window than both GWOL and LWOL. Further experiments show that our scheme can proactively incorporate attacker addresses into the blacklist before these addresses reach the blacklist consumer network, and it can do so in a timely fashion. Finally, our experiments demonstrate that the hit count increase is consistent over time, and the advantages of our blacklist remain stable across various list lengths and testing windows.

The contribution of this paper is the introduction of the highly predictive blacklisting system, which includes our methodology for prefiltering, relevance-based ranking, attacker severity ranking, and final blacklist construction. Ours is the first exploration of a link-analysis-based scheme in the context of security filter production and to quantify the predictive quality of the resulting data. The HPB system is also one of the only new approaches we are aware of for large-scale blacklist publication that has been proposed in many years. However, our HPB system is applicable only to those users who participate as active contributors to collaborative security log data centers. Rather than a detriment, we hope that this fact provides some operators a tangible incentive to participate in security log contributor pools. Finally, the system discussed in this paper, while still a research prototype, has been fully implemented and deployed for nearly a year as a free service on the Internet at DShield.org. Our experience to date leads us to believe that this approach is both scalable and feasible for daily use.

The rest of the paper is organized as follows. Section 2 provides a background on previous work in blacklist generation and related topics. In Section 3 we provide a detailed description of the Highly Predictive Blacklist system. In Section 4 we present a performance evaluation of HPBs, GWOLs, and LWOLS, including assessments of the extent to which the above three desired blacklist properties (hit rate, proactive appearance, and timely inclusion) are realized by these three blacklists. In Section 5 we present a prototype implementation of the HPB system that is freely available to DShield.org log contributors, and we summarize our key findings in Section 6.

2 Related Work

Network address and email blacklists have been around since the early development of the Internet [6]. Today, sites such as DShield regularly compile and publish firewall-parsable filters of the most prolific attack sources reported to its website [17]. DShield represents

a centralized approach to blacklist formulation, providing a daily perspective of the malicious background radiation that plagues the Internet [15, 20]. Other recent examples of computer and network blacklists include IP and DNS blacklists to help networks detect and block unwanted web content, SPAM producers, and phishing sites, to name a few [7, 8, 17, 18]. The HPB system presented here complements, but does not displace these resources or their blacklisting strategies. In addition, HPBs are only applicable to active log contributors (we hope as an incentive), not as generically publishable one-size-fits-all resources.

More agile forms of network blacklisting have also been explored, with the intention of rapidly publishing perimeter filters to control actively spreading malware epidemics [1, 3, 12, 14]. For example, in [14] a peer-to-peer blacklisting scheme is proposed, where each network incorporates an address into its local blacklist when a threshold number of peers have reported attacks from this address. We separate our HPB system from these malware defense schemes. While the HPB system does incorporate a malware-oriented attacker severity metric into its final blacklist selection, we have not contemplated nor propose HPBs for use in the context of dynamic quarantine defenses for malware epidemics.

One key insight that inspired the HPB relevance-based ranking scheme was raised by Katti et al. [10], who identified the existence of stable correlations among the attackers reported by security log contributors. Here we introduce a relevance-based recommendation scheme that selects candidate attack sources based on the attacker overlaps found among peer contributors. This relevance-based ranking scheme can be viewed as a random walk on the correlation graph, going from one node to another following the edges in the graph with the probability proportional to the weight of the graph. This form of random walk has been applied in link-analysis systems such as Google's PageRank [2], where it is used to estimate the probability that a webpage may be visited. Similar link analysis has been used to rank movies [13] and reading lists [19].

The problem of predicting attackers has also been recently considered in [24] using a Guassian process model. However, [24] purely focused on developing statistical learning techniques for attacker prediction based on collaborative filtering. In this paper, we present a comprehensive blacklisting generation system that considers many other characteristics of attackers. The prediction part is only one component in our system. Furthermore, the prediction model presented here is completely different from the one in [24] (Gaussian process model in [24] and link analysis model here). By taking some penalty in predictive power, the prediction model presented here is much more scalable, which is of necessity for implementing a deployable service (Section 5).

Finally, [23] provides a six-page summary of the earliest release of our DShield HPB service, including a high-level description of an early ranking scheme. In this paper we have substantially expanded this algorithm and present its full description for the first time. This present paper also introduces the integration of metrics to capture attack source maliciousness in its final rank selection, and presents the full blacklist construction system. We also present our quantitative evaluation of multiple system properties, and address several open questions that have been raised over the past year since our initial prototype.

3 Blacklisting System

We illustrate our blacklisting system in **Figure 1**. The system constructs blacklists in three stages. First, the security alerts supplied by sensors across the Internet are preprocessed. This removes known noises in the alert collection. We call this the *prefiltering* stage. The preprocessed data are then fed into two parallel engines. One ranks, for each contributors, the attack sources according to their relevance to that contributor. The other scores the sources using a severity assessment that measures their maliciousness. The relevance ranking and the severity score are combined at the last stage to generate a final blacklist for each contributor.

We descibe the prefiltering process in Section 3.1, relevance ranking in Section 3.2, severity score in Section 3.3 and the final production of the blacklists in Section 3.4.

3.1 Prefiltering Logs for Noise Reduction

One challenge to producing high-quality threat intelligence for use in perimeter filtering is that of reducing the amount of *noise* and erroneous data that may exist in the input data that drives our blacklist construction algorithm. That is, in addition to the unwanted port scans, sweeps, and intrusion attempts reported daily within the DShield log data, there are also commonly produced log entries that arise from nonhostile activity, or activity from which useful filters cannot be reliably derived. While it is not possible to separate attack from nonattack data, the HPB system prefilters from consideration logs that match criteria that we have been able to empirically identify as commonly occurring nonuseful input for blacklist construction purposes.

As a preliminary step prior to blacklist construction, we apply three filtering techniques to the DShield alert logs. First, the HPB system removes from consideration DShield logs produced from attack sources from invalid or unassigned IP address space. Here we employ the

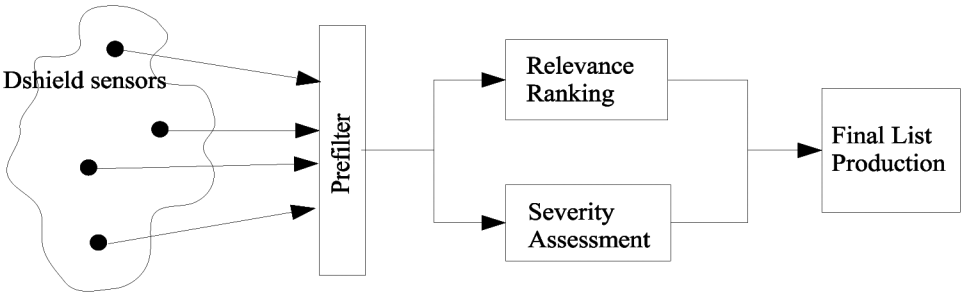

Figure 1: Blacklisting system architecture

bogon list created by the Cymru team that captures addresses that are reserved, not yet allocated, or delegated by the Internet Assigned Number Authority [16]. Typically, such addresses should not be routed, but otherwise do appear anyway in the DShield data. In addition, reserved addresses such as the 10.x.x.x or 192.168.x.x may also appear in misconfigured contributor logs that are not useful for translating into blacklists.

Second, the system prefilters from consideration network addresses from Internet measurement services, web crawlers, or common software update sources. From experience, we have developed a whitelist of highly common sources that, while innocuous from an intrusion perspective, often generate alarms in DShield contributor logs.

Finally, the HPB system applies heuristics to avoid common false positives that arise from commonly 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). Firewalls will commonly time out sessions from these services when the server or client becomes unresponsive or is slow. In practice, the combination of these prefiltering steps provides approximately a 10% reduction in the DShield input stream prior delivery to the blacklist generation system.

3.2 Relevance Ranking

Our notion of attacker relevance is a measure that indicates how close the attacker is related to a particular blacklist consumer. It also reflects the likelihood to which the attacker may come to the blacklist consumer in the near future. Note that this relevance is orthogonal to metrics that measure the severity (or benignness) of the source, which we will discuss in the next section.

In our context, the blacklist consumers are the contributors that supply security logs to a log-sharing repository such as DShield. Recent research has observed the existence of attacker overlap correlations between DShield contributors [10], i.e., there are pairs of contributors that share quite a few common attackers, where the common attacker is defined as a source address that both contributors have logged and reported to the repository. This research also found that this attacker overlap phenomenon is not due to attacks that select targets randomly (as in a random scan case). The correlations are long lived and some of them are independent of address proximity. We exploit these overlap relationships to measure attacker relevance.

We first illustrate a simple concept of attacker relevance. Consider a collection of security logs displayed in a tabular form as shown in **Table 1**. We use the rows of the table to represent attack sources and the columns to represent contributors. We refer to the unique source addresses that are reported within the log repository as *attackers*, and use the terms "attacker" and "source" interchangeably. Since the contributors are also the targets of the logged attacks, we refer to them as *victims*. We will use the terms "contributor" and "victim" interchangeably. 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

Let us assume that **Table 1** represents a series of logs contributed in the recent past by our five victims, v_1 through v_5. Now suppose we would like to calculate the relevance of the sources for contributor v_1 based on these attack patterns. From the attack table we observe

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 does not share attacker overlap 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. But the victims of s_6's attacks share no overlap with v_1. 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.

To formalize the above intuition, we model the attack correlation relationship between contributors using a *correlation graph*, which is a weighted undirected graph $G = (V, E)$. The nodes in the graph consist of the contributors $V = \{v_1, v_2, \ldots\}$. 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 (i.e., occurrences of attacker overlap) between the two corresponding contributors. We now introduce some notation for the relevance model.

Let n be the number of nodes (number of contributors) in the correlation graph. We use $\mathbf{W}$ to denote the adjacency matrix of the correlation graph, where the entry $\mathbf{W}_{(i,j)}$ in this matrix is the weight of the edge between node v_j and v_i. For a source s, we denote by $T(s)$ the set of contributors that have reported an attack from s. $T(s)$ can be written in a vector form $\mathbf{b}^s = \{b_1^s, b_2^s, \ldots, b_n^s\}$ 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 $\mathbf{r}^s = \{r_1^s, r_2^s, \ldots, r_n^s\}$ such that r_v^s is the relevance value of attacker s with respect to contributor v. We use lowercase boldface to indicate vectors and uppercase boldface to indicate matrices. **Table 2** summarizes our notation.

We now describe how to derive the matrix $\mathbf{W}$ from the attack reports. Consider the following two cases. In Case 1, contributor v_i sees attacks from 500 sources and v_j sees 10 sources. Five of these sources attack both v_i and v_j. In Case 2, there are also five common sources. However, v_i sees only 50 sources and v_j sees 10. Although the number of overlapping sources is the same (i.e., 5 common sources), the strength of connection between v_i and v_j is different in the two cases. If a contributor observes a lot of attacks, it is expected that there should be more overlap between this contributor and the others. Let m_i be the number of sources seen by v_i, m_j the number seen by v_j, and m_{ij} the number of common attack sources. The ratio $\frac{m_{ij}}{m_i}$ shows how important v_i is for v_j while $\frac{m_{ij}}{m_j}$ shows how important v_j is for v_i. Since we want $\mathbf{W}_{(i,j)}$ to reflect the strength of the connection between v_i and v_j, we set $\mathbf{W}_{(i,j)} = \frac{m_{ij}}{m_i} \cdot \frac{m_{ij}}{m_j}$. One may view this new $\mathbf{W}$ as a standardized correlation matrix. **Figure 2** shows the matrix $\mathbf{W}$ for **Table 1** constructed using this method.

n	# of contributors
v_i	i-th contributor
$\mathbf{W}$	Adjacency matrix of the correlation graph
$T(s)$	Set of contributors that have reported attack(s) from source s
$\mathbf{b}^s$	Attack vector for source s. $b_i^s = 1$ if $v_i \in T(s)$ and 0 otherwise
$\mathbf{r}^s$	Relevance vector for source s. r_v^s is the relevance value of attacker s with respect to contributor v

Table 2: Summary of Relevance Model Notations

$$\begin{pmatrix} 0 & 0.33 & 0.083 & 0 & 0 \\ 0.33 & 0 & 0.063 & 0 & 0 \\ 0.083 & 0.063 & 0 & 0.13 & 0 \\ 0 & 0 & 0.13 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \end{pmatrix}$$

Figure 2: Standardized Correlation Matrix for Attack Table 1

Given this correlation matrix, we follow the aforementioned intuition and calculate the relevance as $r_i^s = \sum_{j \in T(s)} \mathbf{W}_{(i,j)}$. This is to say that if the repository reports that source s has attacked contributor v_j, this fact contributes a value of $\mathbf{W}_{(i,j)}$ to the source's relevance with respect to the victim v_i. Written in vector form, it gives us

$$\mathbf{r}^s = \mathbf{W} \cdot \mathbf{b}^s. \quad (1)$$

The above simple relevance calculation lacks certain desired properties. For example, the simple relevance value is calculated solely from the observed activities from the source by the repository contributors. In some cases, this observation does not represent the complete view of the source's activity. One reason is that the contributors consist 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. The attacker may be at its early stage or it has attacked many places,

most of which do not participate in the security log sharing system. Therefore, one may want a relevance measure that has a "look-ahead" capability. That is, the relevance calculation should take into consideration possible future observations of the source and include these anticipated observations from the contributors into the relevance values.

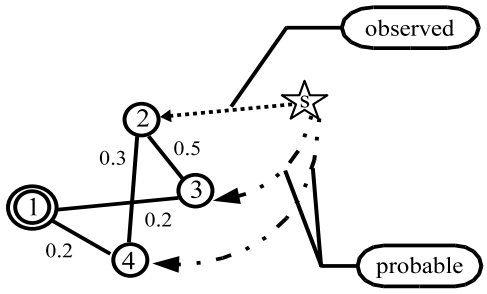

Figure 3: Relevance Evaluation Considers Possible Future Attacks

Figure 3 gives an example where one may apply this "look-ahead" feature. (Examples here are independent of the one shown in **Table 1**.) The correlation graph of **Figure 3** consists of four contributors numbered 1, 2, 3, and 4. Contributor 2 reported an attack from source s (represented by the star). Our goal is to evaluate how relevant this attacker is to contributor 1 (double-circled node). Using **Equation 1**, the relevance would be zero. However, we observe that s has relevance 0.5 with respect to contributor 3 and relevance 0.3 with respect to contributor 4. Although at this time, contributors 3 and 4 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 1, contributors 3 and 4 pass to contributor 1 their relevance values after multiplying them with the weights on their edges, respectively. The attacker's relevance value for contributor 1 then is 0.5*0.2+0.3*0.2 = 0.16. Note that, had s actually attacked contributors 3 and 4, the contributors would have passed the relevance value 1 (again after multiplying them with the weights on the edges) to contributor 1.

This can be viewed as a relevance propagation process. 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.

This relevance propagation process has another benefit besides the "look-ahead" feature. Consider the correlation graph given in **Figure 4** (a). The subgraph formed by nodes 1, 2, 3, and 4 is very different from that formed by nodes 1, 5, 6, and 7. The subgraph from nodes 1, 2, 3, and 4 is well connected (in fact it forms a clique). The contributors in the subgraph are thus more tied together. We call them a *correlated group*. (We use a dotted circle to indicate the correlated group in **Figure 4**.) There may be certain intrinsic similarities between the members in the correlated group (e.g., IP address proximity, similar vulnerability). Therefore, it is natural to assign more relevance to source addresses that have attacked other contributors in the same correlated group. For example, consider the sources s and s' in **Figure 4**. They both attacked three contributors. All the edges in the correlation graph have the same weights. (Hence, we omitted the weights in the figure.) We would like to say that s is more relevant than s' for contributor 1. If we calculate the relevance value by **Equation 1**, the values would be the same for the two attackers. Relevance propagation helps to give more value to the attacker s because members of the correlated group are well connected. There are more paths in the subgraph that lead from the contributors where the attack happened to the contributor for which we are evaluating the attacker relevance. For example, the relevance from contributor 2 can propagate to contributor 3 and then to contributor 1. It can also go to contributor 4 and then to contributor 1. This is effectively the same as having an edge with larger weight between the contributors 2 and 1. Therefore, relevance propagation can effectively discover and adapt to the structures in the correlation graph. The relevance values assigned then reflect certain intrinsic relationships among contributors.

We extend **Equation 1** to employ relevance propagation. If we propagate the relevance values to the immediate neighbors in the correlation graph, we obtain a relevance vector $\mathbf{W} \cdot \mathbf{b^s}$ that represents the propagated values. Now we propagate the relevance values one more hop. This gives us $\mathbf{W} \cdot \mathbf{W} \cdot \mathbf{b^s} = \mathbf{W}^2 \cdot \mathbf{b^s}$. The relevance vector that reflects the total relevance value each contributor receives is then $\mathbf{W} \cdot \mathbf{b^s} + \mathbf{W}^2 \cdot \mathbf{b^s}$. If we let the propagation process iterate indefinitely, the relevance vector would become $\sum_{i=1}^{\infty} \mathbf{W}^i \cdot \mathbf{b^s}$. There is a technical detail in this process we need to resolve. Naturally, we would like the relevance value to decay along the path of propagation. The further it goes on the graph, the smaller its contribution becomes. To achieve this, we scale the matrix $\mathbf{W}$ by a constant $0 < \alpha < 1$ such that the 2-norm of the new matrix $\alpha\mathbf{W}$ becomes smaller than one. With this modification, an attacker will have

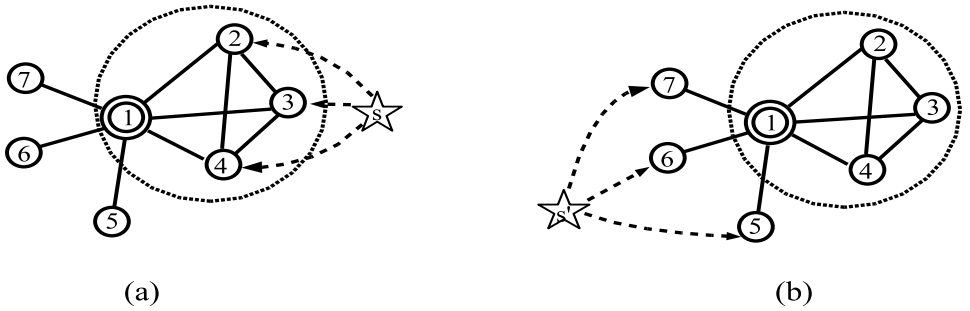

Figure 4: Attacks on Members in a Correlated Group Contribute More Relevance

only a negligible relevance value to contributors that are far away in the correlation graph. Putting the above together, we compute the relevance vector by the following equation:

$$\mathbf{r}^s = \sum_{i=1}^{\infty} (\alpha \mathbf{W})^i \cdot \mathbf{b^s} \tag{2}$$

We observe that $\mathbf{b^s} + \mathbf{r}^s$ is the solution for $\mathbf{x}$ in the following system of linear equations:

$$\mathbf{x} = \mathbf{b^s} + \alpha \mathbf{W} \cdot \mathbf{x} \tag{3}$$

The linear system described by **Equation 3** is exactly the system used by Google's PageRank [2]. PageRank analyzes the link structures of webpages to determine the relevance of each webpage with respect to a keyword query. In PageRank, b^s is set to be an all-one vector and $\mathbf{W}$ is determined by letting $\mathbf{W}_{(i,j)}$ be 1/(# of outgoing links on page j) if one of these outgoing links points to webpage i, and $\mathbf{W}_{(i,j)} = 0$ otherwise. Therefore, PageRank propagates relevance where every node provides an initial relevance value of one. In our relevance calculation, only nodes whose corresponding contributors have reported the attacker are assigned one unit of initial relevance. Similar to the PageRank values that reflect the link structures of the webpages, our relevance values reflect the structure of the correlation graph that captures intrinsic relationships among the contributors.

Equation 3 can be solved to give $\mathbf{x} = (\mathbf{I} - \alpha \mathbf{W})^{-1} \cdot \mathbf{b}^s$, where $\mathbf{I}$ is the identity matrix. Also, since $\mathbf{x} = \mathbf{r}^s + \mathbf{b}^s$, $\mathbf{r}^s = (\mathbf{I} - \alpha \mathbf{W})^{-1} \cdot \mathbf{b}^s - \mathbf{b}^s = [(\mathbf{I} - \alpha \mathbf{W})^{-1} - \mathbf{I}] \cdot \mathbf{b}^s$. This gives the relevance vector for each attack source. The sources are then ranked, for each contributor, according to the relevance values. As each attack source has a potentially different relevance value for each contributor, the rank of a source with respect to different contributors is different. Note that our concept of relevance measure and relevance propagation does not depend on a particular choice of the $\mathbf{W}$ matrix. As long as $\mathbf{W}$ reflects the connection weight between the contributors, our relevance measure applies.

3.3 Analyzing Attack Pattern Severity

We now consider the problem of measuring the degree to which each attack source exhibits known patterns of malicious behavior. In the next section, we will disuss how this measure can be fused into our final blacklist construction decisions. In this section we will describe our model of malicious behavior and the attributes we extract to map each attacker's log production patterns to this model.

Our model of *malicious behavior*, in this instance, focuses on identifying typical scan-and-infect malicious software (or malware). We define our malware behavior pattern as that 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. This behavior pattern matches the malware behavior pattern documented by Yegeneswaren et.al. in [20], as well as our own most recent experiences (within the last twelve months) of more than 20K live malware infections observed within our honeynet [21]. Other potential malware behavior patterns may be applied, for example, such as the scan-oriented malicious address detection schemes outlined in the context of dynamic signature generation [11] and malicious port scan analysis [9]. 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.

For the set of log entries over the relevance-calculation time window, we calculate several attributes for each attacker's /24 network address. (Our blacklists are specified on a per /24 basis, meaning that a single malicious address has the potential to induce a LAN-wide filter. This is standard practice for DShield and other blacklists.) For each attacker, we assign a score to target ports associated with the attacker, assigning a different weight depending on whether or not the port is associated with known malware communications.

Let MP be the set of malware-associated ports, where we currently uses the definition in Figure 5. This MP

$$\left(\begin{array}{llllll} 53-UDP & 69-UDP & 137-UDP & 21-TCP & 53-TCP & 42-TCP \\ 135-TCP & 139-TCP & 445-TCP & 559-TCP & 1025-TCP & 1433-TCP \\ 2082-TCP & 2100-TCP & 2283-TCP & 2535-TCP & 2745-TCP & 2535-TCP \\ 3127-TCP & 3128-TCP & 3306-TCP & 3410-TCP & 5000-TCP & 5554-TCP \\ 6101-TCP & 6129-TCP & 8866-TCP & 9898-TCP & 10000-TCP & 10080-TCP \\ 12345-TCP & 11768-TCP & 15118-TCP & 17300-TCP & 27374-TCP & 65506-TCP \\ 4444-TCP & 9995-TCP & 9996-TCP & 17300-TCP & 3140-TCP & 9033-TCP \\ 1434-UDP & & & & & \end{array}\right)$$

Figure 5: Malware Associated Ports

is derived from various AV lists and our honeynet experiences. We do not argue that this list is complete and can be expanded across the life of our HPB service. However, our experiences in live malware analysis indicate that the entries in MP are both highly common and highly indicative of malware propagation.

Let the number of target ports that attacker s connects to be c_m, and the total number of unique ports connected to be defined as c_u. We associate a weighting (or importance) factor w_m for all ports in MP, and a weighting factor w_u for all nonmalware ports. We then compute a malware port score (PS) metric for each attacker as follows:

$$PS(s) = \frac{(w_u \times c_u) + (w_m \times c_m)}{c_u} \quad (4)$$

Here, we intend w_m to be of greater weight than w_u, and choose an initial default of $w_m = 4 * w_u$. PS has the property that even if a large c_m is found, if c_u is also large (as in a horizontal portscan), then PS will remain small. Again, our intention is to promote a malware behavior pattern in which malware propagation will tend to target fewer specific ports, and is not associated with attackers that engage in horizontal port sweeps.

Next, we compute the set of unique target IP addresses connected to by attacker s. We refer to this count as $TC(s)$. A large TC represents confirmed IP sweep behavior, which we strongly associate with our malware behavior model. TC is *the* exclusive prioritization metric used by GWOL, whereas here we consider TC a secondary factor to PS in computing a final malware behavior score. We could also include metrics regarding the number of DShield sensors (i.e., unique contributor IDs) that have reported the attacker, which arguably represents the degree of *consensus* in the contributor pool that the attack source is active across the Internet. However, the IP sweep pattern is of high interest, even when the IP sweep experiences may have been reported only by a smaller set of sensors.

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, Russian, the Czech Republic) that exclusively target international victims. However, this may also illustrate a weakness in the DShield contributor pool, as there may be very few contributors that operate sensors within these countries. We incorporate a dampening factor δ ($0 \leq \delta \leq 1$) that allows the consumer to express the degree to which the IR factor should be nullified in computing the final severity score for each attacker.

Finally, we compute a malware severity score $MS(s)$ for each candidate attacker that may appear in the set of final blacklist entries:

$$MS(s) = PS(s) + \log(TC(s)) + \delta \log(IR(s)) \quad (5)$$

The three factors are computed in order of significance in mapping to our malware behavior model. Logarithm is used because in our model, the secondary metric (TC) and the tertiary metric (IR) are less important than the malware port score and we only care about their order of magnitude.

3.4 Blacklist Production

For each attacker, we now have both its relevance ranking and its severity score. We can combine them to generate a final blacklist for each contributor.

For the final blacklist, we would like to include the attackers that have strong relevance and discard the nonrelevant attackers. To generate a final list of length L, we use the attacker's relevance ranking to compile a candidate list of size $c \cdot L$. (We often set $c = 2$.) Then, we use severity scores of the attackers on the candidate list to adjust its ranking and pick the L highest-ranked attackers to form the final list. Intuitively, the adjustment should promote the rank of an attacker if the severity assessment indicates that it is very malicious. Toward this goal, we define a final score that combines the attacker's relevance rank in the candidate list and its severity assessment. In particular, let k be the relevance rank of the attacker s

(i.e., s is the k-th entry in the candidate list). Recall from last section $MS(s)$ is the severity score of s. The final score $fin(s)$ is defined to be

$$fin(s) = k - \frac{L}{2} \cdot \Phi(MS(s)) \quad (6)$$

where

$$\Phi(x) = \frac{1}{2}(1 + erf(\frac{x-\mu}{d}))$$

where $erf(\cdot)$ is the "S" shaped Gaussian error function. We plot $\Phi(x)$ in **Figure 6** with $\mu = 4$ and different d.

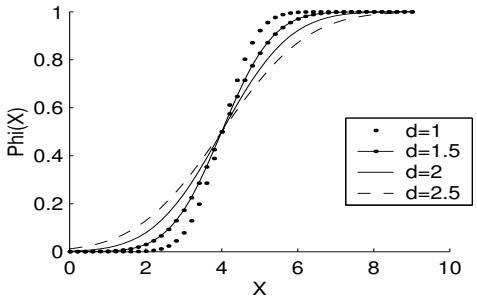

Figure 6: Phi with different d value

$\Phi(MS(s))$ promotes the rank of an attacker according to its maliciousness. The larger the value of $\Phi(MS(s))$ is, the more the attacker is moved above comparing to its original rank. A $\Phi(MS(s))$ of value 1 would then move the attacker above for one half of the size of the final list comparing to its original rank. The "S" shaped $\Phi(\cdot)$ transforms the severity assessment $MS(s)$ into a value between 0 and 1. The less-malicious attackers often give an assessment score below 3. After transformation, they will receive only small promotions. On the other hand, malicious attackers that give an assessment score above 7 will be highly promoted.

To generate the final list, we sort the $fin(s)$ values of the attackers in the candidate list and then pick L of them that have the smallest $fin(s)$.

4 Experiment Results

We created an experimental HPB blacklist formulation system. To evaluate the HPBs, 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. Since our relevance measure is based on correlations between contributors, HPB production is not applicable to contributors that have submitted very few reports (DShield has contributors that hand-select or sporadically contribute logs, providing very few alerts). We therefore exclude those contributors that we find effectively have no correlation with the wider contributor pool or simply have too few alerts to produce meaningful results. For this analysis, we found that we could compute correlation relationships for about 700 contributors, or 41% of the DShield contributor pool.

To assess the performance of the HPB system, we compare its performance relative to the standard DShield-produced GWOL [17]. In addition, we compare our HPB performance to that of LWOLs, which we compute individually for all contributors in our comparison set. For the purpose of our comparative assessment, we fixed the length of all three competing blacklists to exactly 1000 entries. However, after we present our comparative performance results, we will then continue our investigation by analyzing how the blacklist length affects the performance of the HPBs.

In the experiments, 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. We call the period used for producing blacklists the *training window* and the period for testing the *prediction window*. In practice, the training period represents a snapshot of the most recent history of the repository, used to formulate each blacklist for a contributor that is then expected to use the blacklist for the length of the prediction window. The sizes of these two windows are not necessarily equal. We will first describe experiments that use 5-day lengths for both the training window and the prediction window. We then present experiments that investigate the effects of the two windows' lengths on HPB quality.

4.1 Hit Count Improvement

DShield logs submitted during the prediction window are used to determine how many sources included within a contributor's HPB are indeed encountered during that prediction window. We call this value the blacklist *hit count*. We view each blacklist address filter not encountered by the blacklist consumer as an *opportunity cost* to have prevented the deployment of other filters that could have otherwise blocked unwanted traffic. In this sense, we view our hit count metric as an important measure of the effectiveness of a blacklist formulation algorithm. Note that our HPBs are formulated with severity analysis while the other lists are not. As the severity analysis prefers malicious activities, we expect that the hits on the HPBs are more malicious.

To compare the three types of lists, we take 60 days of data, divided into twelve 5-day windows. We repeat the experiment 11 times using the i-th window as the training window and the $(i+1)$-th window as the testing window. In the training window, we construct HPB, LWOL, and

Window	GWOL total hit	LWOL total hit	HPB total hit	HPB/GWOL	HPB/LWOL
1	81937	85141	112009	1.36701	1.31557
2	83899	74206	115296	1.37422	1.55373
3	87098	96411	122256	1.40366	1.26807
4	80849	75127	115715	1.43125	1.54026
5	87271	88661	118078	1.353	1.33179
6	93488	73879	122041	1.30542	1.6519
7	100209	105374	133421	1.33143	1.26617
8	96541	91289	126436	1.30966	1.38501
9	94441	107717	128297	1.35849	1.19106
10	96702	94813	128753	1.33144	1.35797
11	97229	108137	131777	1.35533	1.21861
Average	90879 ± 6851	90978 ± 13002	123098 ± 7193	1.36 ± 0.04	1.37 ± 0.15

Table 3: Hit Number Comparison between HPB, LWOL and GWOL

	Contributor Percentage	Average Increase	Median Increase	StdDev	Increase Range
Improved vs. GWOL	90%	51	22	89	1 to 732
Poor vs. GWOL	7%	-27	-7	47	-1 to -206
Improved vs. LWOL	95%	75	36	90	1 to 491
Poor vs. LWOL	4%	-19	-9	28	-1 to -104

Table 4: Hit Count Performance, HPB vs. (GWOL and LWOL), Length 1000 Entries

GWOL. Then the three types of lists are tested on the data in the testing window.

Table 3 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. We see that in every window, HPB has more hits than GWOL and LWOL. Overall, HPBs predict 20-30% more hits than LWOL and GWOL. Note that there are quite large variances among the number of hits between time windows. Most of the variances, however, are not from our blacklist construction, rather they are from the variance among the number of attackers the networks experience in different testing windows.

	Increase Average	Increase Median	Increase StdDev	Increase Range
vs. GWOL	129	78	124	40 to 732
vs. LWOL	183	188	93	59 to 491

Table 5: Top 200 Contributors' Hit Count Increases (Blacklist Length 1000)

The results in Table 3 show HPB's hit improvement over time windows. We now investigate the distribution of the HPB's hit improvement across contributors in one time window. We use two quantities for comparison. The first is the hit count improvement, which is simply the HPB hit count minus the hit count of the other list. The second comparative measure we used is the relative hit count improvement (RI), which is the ratio in percentage of the HPB hit count increase over the other blacklist hit count. If the other list hit count is zero we define RI to be 100x the HPB hit count, and if both hit counts are zero we set RI to 100.

Table 5 provides a summary of hit-count improvement for the 200 contributors where HPBs perform the best. The hit-count results for all the contributors are summarized in **Table 4**.

Figure 7 compares HPB to GWOL. The left panel of the figure plots the histogram showing the distribution of the hit improvement across the contributors. The x-axis indicates improvements, and the hight of the bars represents the number of contributors whose improvement fall in the corresponding bin. Bars left to $x = 0$ represent contributors for whom the HPB has worse performance and bars on the right represent contributors for whom HPBs performed better. For most contributors, the improvment is positive. The largest improvement reaches 732. For only a few contributors, HPB performs worse in this time window.

The panel on the right of **Figure 7** plots the RI (ratio % of HPB's hit count increase over GWOL's hit count) distribution. We sort the RI values and plot them against the contributors. We label the x-axis by cummulative percentage, i.e., a tick on x-axis represents the percentage of contributors that lie to the left of the tick. For example, the tick 20 means 20 percent of the contributors lie left to this tick. There are contributors for which the RI value can be more than 3900. Instead of showing such large RI values, we cut off the plot at RI value 300. From the plot, we see that 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 con-

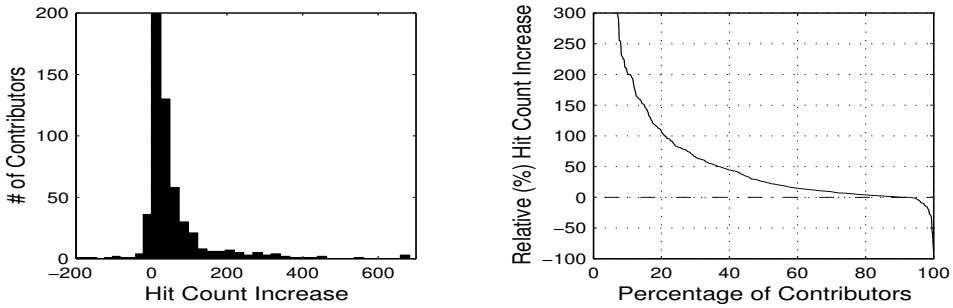

Figure 7: Hit Count Comparison of HPB and GWOL: Length 1000 Entries

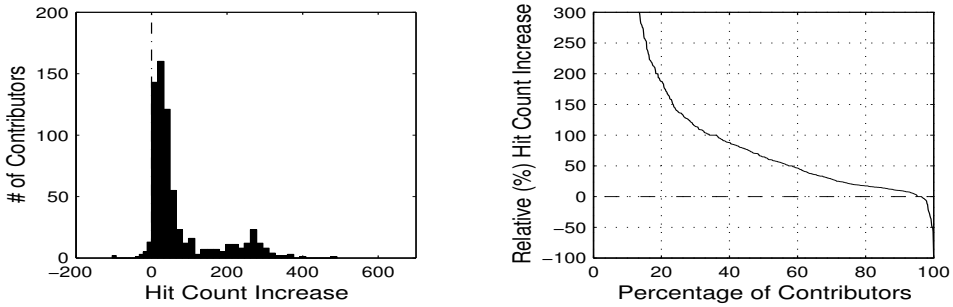

Figure 8: Hit Count Comparison of HPB and LWOL: Length 1000 Entries

tributors (about 7%), HPBs perform worse. (We discuss the reasons why HPB may perform worse in Section 4.4.)

Figure 8 compares HPB hit counts to those of LWOL. The data are plotted in the same way as in **Figure 7**. Overall, HPBs demonstrate a performance advantage over LWOL. The IV and RI values also exhibit similar distributions. However, comparing **Figures 8** and **7**, we see that HPB has more hit improvement comparing to LWOL than to GWOL in this time window.

4.2 Prediction of New Attacks

One clear motivating assumption in secure collaborative defense strategies is that participants have the potential to prepare themselves from attacks that they have not yet encountered. We will say that a *new attack* occurs when a contributor produces a DShield log entry from a source that this contributor has never before reported. In this experiment, we show that HPB analysis provides contributors a potential to predict more new attacks than GWOL. (LWOL is not considered, since by definition it includes *only* attackers that are actively hitting the LWOL owner.) For each contributor, we construct two new HPB and GWOL lists with equal length of 1000 entries, such that no entries have been reported by the contributor during our training window. We call these lists HPB-local (HPB minus local) and GWOL-local (GWOL minus local), respectively. **Figure 9** compares HPB-local and GWOL-local on their ability to predict on new attack sources for the local contributor. These hit number plots demonstrate that HPB-local provides substantial improvement over the predictive value of GWOL.

4.3 Timely Inclusion of Sources

By timely inclusion, we refer to the ability of a blacklist to incorporate addresses relevant to the blacklist owner *before* those addresses have saturated the Internet. To investigate the timeliness of the GWOL, LWOL, and the HPB we examine how many contributors need to report a particular attacker before it can be included into the respective blacklists. We focus our attention on the set of attackers within these blacklists that *did* carry out attacks during the prediction window. And we use the number of distinct victims (contributors) that a source attacked in the training window to measure the extent to which the source has saturated the Internet. **Figure 10** plots the distribution of the number of distinct victims across different attackers on the three blacklists. As expected, the attackers that get selected on the GWOL were the most prolific in the training period. In particular, all the sources on the GWOL have attacked more than 20 contributors and almost 1/3 of them attacked more than 200 contributors. To some extent, these attackers have saturated the Internet with their activities. (DShield sensors are a very small sample of the Internet. A random at-

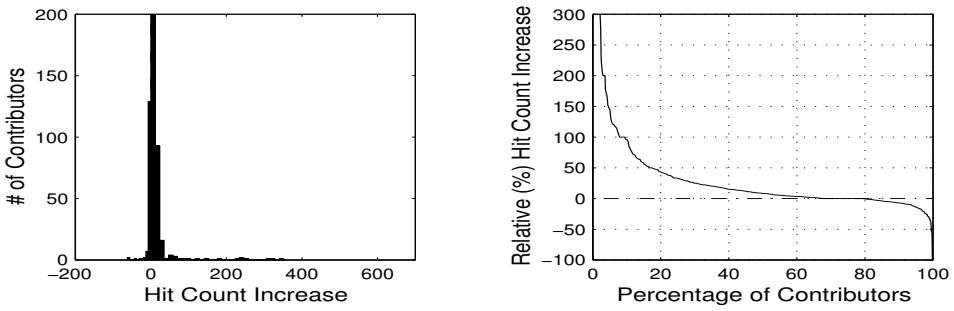

Figure 9: HPB-local Predicts More New Attacks Than GWOL-local

tacker has to target many places to be picked up by the sensors.) The LWOLs select attacker addresses that focused on the local networks. Most of these addresses had attacked far fewer contributors. HPBs's distribution is close to that of the LWOL, hence allowing the incorporation of attackers that have not saturated the Internet.

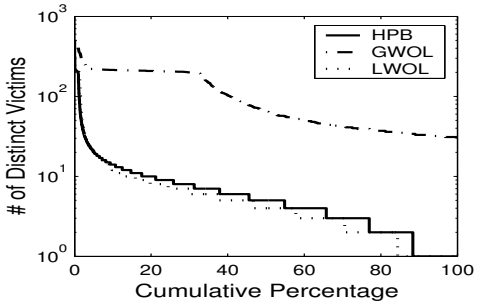

Figure 10: Cumulative Distribution of Distinct Victim Numbers

4.4 Performance Consistency

The results in the above experiments show that the HPB provides an increase in hit count performance across the majority of all contributors. We now ask the following question: is the HPB's performance consistent for a given contributor over time? In this experiment, we investigate this consistency question.

We use a 60-day DShield dataset. We divide it into 12 time windows, $T_0, T_1, \ldots, T_{11}$. We generate blacklists from data in time window T_{i-1} and test the lists on data in T_i. For each contributor v, we compare HPB with GWOL and obtain eleven improvement values for window T_0 to T_{10}. We denote them
$IVs(v) = \{IV_0(v), IV_2(v), \ldots IV_{10}(v)\}$. We then define a consistency index (CI) for each contributor. If $IV_i(v) \geq 0$, we say that the HPB performs well for v in window i. Otherwise, we say that the HPB performs worse. CI is the difference between the number of windows in which HPB performs well and the ones in which HPB performs poorly, i.e., $CI(v) = |\{p \in IVs(v) : p \geq 0\}| - |\{p \in IVs(v) : p < 0\}|$. If HPB consistently performs better than GWOL for a contributor, its $CI(v)$ should be close to 11. If it consistently performs worse, the CI value will be close to -11. However, if the HPB performance flip-flops, its CI value will be close to zero. **Figure 11** plots the sorted CI values against the contributors. (Again, we label the x-axis by cummulative percentage.) We see that for almost 70% of the contributors, HPB's performance is extremely consistent. They all have a CI value of 11, meaning for the eleven time windows, the HPB always predicts more hits for them than GWOL. For more than 90% of the contributors, HPBs demonstrate fairly good consistency. With few contributors does the performance switch back and forth. Only 5 contributors show performance index below -3.

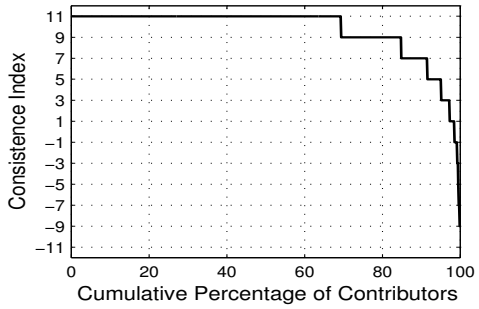

Figure 11: Cumulative Distribution of Consistency Index

The consistency investigation sheds some light on the reason why there is a small percentage of contributors for which the HPBs (sometimes) perform worse than the other list. HPB construction is based on the relevance measure. The relevance relates attack sources to contributors according to the past security logs collected by the repository. If a contributor has relatively stable correlations (stable for several days) with other contributors or it experiences stable attack patterns, the relevance measure can capture this and thus produce blacklists with more

hits. Such HPBs will also be consistent in hit-count performance. On the other hand, if the correlation is not stable or the attacks exhibit few patterns, the relevance measure will be less effective and may produce blacklists with fewer hits. Such HPBs will not be consistent in performance because sometimes they may guess right and produce more hits and sometimes they may guess wrong.

This can be seen in **Figure 11**. All the consistent HPBs have CI value 11. These HPBs have both consistency and better hit-count performance. There is no HPB that shows CI value -11. HPB never performs consistently worse.

This is particularly useful because the consistency of an HPB's performance can be used to indicate whether the HPB user (the contributor) has stable correlations. If so, HPBs can be better blacklists to use. The experiment result suggests that most of the contributors have stable correlations. In practice, given a few cycles of computing HPB and GWOL for a DShield contributor, we can provide an informed recommendation as to which list that contributor should adopt over a longer term.

4.5 Blacklist Length

In this experiment, we vary the length of the blacklists to be 500, 1000, 5000 and 10000. We then compare the hit counts of HPBs, GWOLs, and LWOLs. Because in all the experiments, the improvements for different contributors display similar distributions, we will simply plot the medians of the hit rates of these respective blacklists. (Hit rate is the hit count divided by the blacklist length.) Our results are illustrated in **Figure 12**, and show that HPBs have the hit rate advantage for all these choices in blacklist length. The relative amount of advantage is also maintained across different lengths.

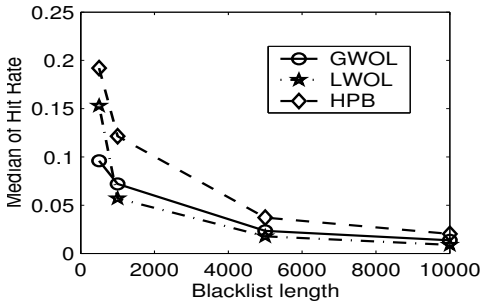

Figure 12: Hit Rates of HPB, GWOL, and LWOL with Different Lengths

Although the hit rate for the shorter lists is higher, the number of hits are larger for the longer lists. This is so for all three types of blacklists. It shows that the longer the list is, the more entries on the list are wasted (in the sense that they do not get hit). Therefore, it may not always be desirable to use very long lists.

4.6 Training and Prediction Window Sizes

We now investigate how far into the future the HPB can maintain its advantage over GWOL and LWOL, and how different training window sizes affect an HPB's hit count. The former helps to determine how often we need to recompute the blacklist, and the latter helps to select the right amount of history data as the input to our system. The left panel of **Figure 13** shows the median of the hit count of HPB, GWOL, and LWOL on day $1, 2, 3, \ldots, 20$ for each individual day in the prediction window. All lists are generated using data from a 5-day window prior to the prediction window. For all blacklists, the number of hits decreases along time. The HPB maintains an advantage over the entire duration of the prediction window. From this plot, we also see that the blacklists need to be refreshed frequently. In particular, there may be an almost 30% hit drop when the HPB is more than a week old.

The right panel of **Figure 13** plots hit-number medians for four HPBs. These HPBs are generated in a slightly different way from the HPBs we used so far. In previous experiments, to generate an HPB, we produce the correlation matrix from a set of attack reports. Then the sources in the same set of reports are selected into HPBs based on their relevance. In this experiment, we construct the correlation matrix using reports from training windows of size 2, 5, 7, and 10 days. Then the sources that are in the reports within the 5-day window right before the prediction (test) window are picked based on their relevance. In this formulation, we exclude sources that appear only in reports from distant history; we view their extended silence to represent a significant loss in relevance. The remainder of the test is performed in the same way as the previous experiments, i.e., the hit counts are obtained in the following 5-day prediction window. The experiment result shows that there is a slight increase in the hit counts going from a 2-day training window to a 5-day training window. The hit counts then remain roughly the same for the other training-window size. This indicates that for most of the contributors, the correlation matrix can be quite stable over time.

5 An Example Blacklisting Service

In mid 2007, we deployed an initial prototype implementation of the HPB system, providing a subset of the features described in this paper. This initial deployment was packaged as a free Internet blacklisting service for DShield log contributors [22,23]. HPB blacklists

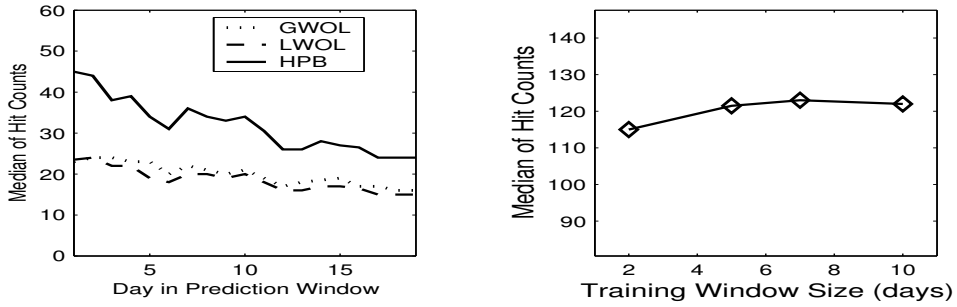

Figure 13: Effect of Training Window and Prediction Window Size on HPB's hit count

are constructed for all contributors daily, and each contributor can download her individual HPB through her DShield website account. To date, we have had a relative small pool of HPB downloaders (roughly 70 users over the most 3 months). We now describe several aspects of fielding a practical and scalable implementation of an HPB system based on our initial deployment experiences. We present an assessment of the algorithm complexity, the DShield service implementation, and discuss some open questions raised from the open release of our service.

5.1 Algorithm Complexity

Because HPBs are constructed from a relatively high-volume corpus of security logs, our system must be prepared to process well over 100M log entries per day to process entries over the current 5-day training window. The bottleneck of the system is the relevance ranking. Therefore, our complexity discussion focuses on the ranking algorithm. There is always an amount of complexity that is linear to the size of the alert data. That is, let $N(data)$ be the number of alerts in the data collection; we have a minimum complexity of $O(N(data))$. Our discussion will focus on other complexities incurred by the algorithm besides this linear-time requirement.

We denote by $N(s)$ and $N(v)$ the number of sources in the data collection and the number of contributors to the repository respectively. In practice, one can expect $N(v)$ to be in the order of thousands while $N(s)$ is much larger, typically in the tens of millions. We obtain $\mathbf{W}$ and $\mathbf{b}^s$ by going through the repository and doing simple accounting. The adjacency matrix $\mathbf{W}$ requires the most work to construct. To obtain this matrix, we record every overlapped attack while going through the alert data and then perform standardization. The latter steps require us to go through the whole matrix, which results in $O(N(v)^2)$ complexity.

Besides going through the data, the most time-consuming step in the relevance estimate process is the computation that solves the linear equations in **Equation 3**. At first glance, because for each source s, we have a linear system determined by **Equation 3**, it seems that we need to solve $N(s)$ linear systems. This can be expensive as $N(s)$ is very large. Further investigation shows that while $\mathbf{b}^s$ is different per source s, the $(\mathbf{I}-\mathbf{W})^{-1}$ part of the solution to **Equation 3** is the same for all s. Therefore, we need to compute it only once, which requires $O(N(v)^3)$ time by brute force or $O(N(v)^{2.376})$ using more sophisticated methods [5]. Because $\mathbf{b}^s$ is sparse, once we have $(\mathbf{I}-\mathbf{W})^{-1}$, the total time to obtain the ranking scores for all the sources and all the contributors is $O(N(v)\cdot N(data))$. Assuming $N(v)^2$ is much smaller than $N(data)$, the total complexity to make relevance ranking is $O(N(v)\cdot N(data))$. For a data set that contains a billion records contributed by a thousand sensors, generating a thousand rankings requires only several trillion operations (additions and multiplications). This can be easily handled by modern computers. In fact, in our experiments, with $N(data)$ in the high tens of millions and $N(v)$ on the order of one thousand, it takes less than 30 minutes to generate all contributor blacklists on an Intel Xeon 3.6 GHz machine.

5.2 The DShield Implementation

The pragmatics of deploying an HPB service through the DShield website are straightforward. DShield log contributors are already provided private web accounts in order to review their reports. However, to ease the automatic retrieval of HPBs, users are not required to log in via DShield's standard web account procedure. Instead, contributors wishing to access their individual HPBs can create account-specific hexadecimal tokens, and can then append this token to the HPB URL. This token has a number of advantages, particularly for developing and maintaining automated HPB retrieval scripts. That is, a user account password may be changed regularly, but the retrieval token (and script) will remain unaffected.

To provide further protection of the integrity and confidentiality of an HPB the user may also pull the HPB via

```
# DShield Customized Blocklist
# created 2007-01-19 12:13:14 UTC
# for userid 11111
# some rights reserved,  DShield Inc.,  Creative Commons Share Alike License
# License and Usage Info: http://www.dshield.org/blocklist.html
1.1.1.1      255.255.255.0        test network
2.2.2.2      255.255.255.0        another test.  This network does not exist
# End of list
```

Figure 14: A Sample Blocklist from DShield Implementation

https. A detached PGP signature can be retrieved in case https is not available or not considered a sufficient proof of authenticity.

HPBs are distributed using a simple tab-delimited format. The first column identifies the network address. The second column provides the netmask. Additional columns are used to provide more information about the respective offender, such as the name of the network and country of origin (or type of attacks seen). These additional columns are intended for human review of the HPB. Comments may be added to the blocklist. All comments start with a # mark. A sample blocklist is shown in Figure 14.

5.3 Gaming the System

As we have made efforts to implement, test, and advertise early versions of the HPB system, several open questions have been raised regarding the ability of adversaries to *game* the HPB system. That is, can an attacker contribute data to DShield with the intention of manipulating HPB production in ways that negatively harm HPB quality? Let us consider several questions that arise from the fact that HPBs are derived from volunteer sources, which may include dishonest contributors that are actively trying to harm or negatively manipulate HPB results.

Can an attacker cause a consumer to incorporate an unsuspecting victim address into a third party's HPB? Let us assume that attacker A participates as one or more DShield contributors (A might register multiple IDs) and knows that consumer C is also a DShield contributor and an active HPB user. Furthermore, A would like to cause address B to be inserted into consumer C's HPB. There are two potential strategies A can pursue to achieve this goal. First, A can spoof attacks as address B, directing these attacks to other contributors that are highly correlated with A. However, C's correlated contributor set is neither readily available to A (unless A is a DShield insider) or necessarily stable over time. More plausibly, A could artificially cause his own contributor IDs to report the same attacks as C. He can do this by attacking C with a set of spoofed addresses, and then reporting similarly spoofed logs from his contributor IDs. Once a sufficient set of attack logs with identical spoofed attackers is reported by C and A, C could then positively influence the likelihood that address B will be inserted into A's HPB. While this is a possible threat, we also observe that similar attacks can be launched against GWOL and more trivially against LWOL. Furthermore, in the case of GWOL, B will be inserted in **all** consumers' GWOLs, whereas A must launch this attack individually against each HPB consumer.

Can an attacker cause his own address to be excluded from a specific third-party HPB? Let us assume that A would like to guarantee that address B will not appear in C's HPB. This is very difficult for A to guarantee. While A may cause artificial alignment between his and C's logs using the alert spoofing method discussed above, A cannot control what other addresses may also align with C. If B attacks other contributors that are aligned with C, B has the potential to enter C's HPB.

Can an attacker fully prevent or poison all HPB production? In short, yes. Data poisoning is a fundamental threat that arises in all volunteer contributor-based data centers, and is an inherently difficult threat to overcome. However, DShield does occasionally experience, and incorporate countermeasures for issues such as *accidental* flooding and sensor misconfiguration. DDoS threats also arise and are dealt with by DShield case by case.

HPB generation could also be specifically targeted by a malicious contributor that attempts to artificially inflate the number of attacker or victim addresses, which will increase the values of s or v, as described in our complexity analysis, Section 5.1. However, to sufficiently prohibit HPB production, the contributor would necessarily produce highly anomalous volumes of attackers (or sources) that would likely allow us to identify and (temporarily) filter this contributor.

6 Conclusion

In this paper, 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 pre-

filtering 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, better *new attacker* prediction quality, and long-term performance stability.

In April of 2007, we released a highly predictive blacklist service at DShield.org. We view this service as a first experimental step toward a new direction of high-quality blacklist generation. We also 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.

7 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] ANAGNOSTAKIS, K. G., GREENWALD, M. B., IOANNIDIS, S., KEROMYTIS, A. D., AND LI, D. A cooperative immunization system for an untrusting Internet. In *Proceedings of the 11th IEEE International Conference on Networks (ICON'03)* (October 2003).

[2] BRIN, S., AND PAGE, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems 30*, 1-7 (1998), 107–117.

[3] CAI, M., HWANG, K., KWOK, Y., SONG, S., AND CHEN, Y. Collaborative Internet worm containment. *IEEE Security and Privacy Magazine 3*, 3 (May/June 2005), 25–33.

[4] CHEN, Z., AND JI, C. Optimal worm-scanning method using vulnerable-host distributions. *International Journal of Security and Networks (IJSN) Special Issue on Computer & Network Security 2*, 1 (2007).

[5] COPPERSMITH, D., AND WINOGRAD, S. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation 9* (1990), 251–280.

[6] HUMPHRYS, M. The Internet in the 1980s. `http://www.computing.dcu.ie/~humphrys/net.80s.html`, 2007.

[7] INCORPORATED, G. List of blacklists. `http://directory.google.com/Top/Computers/Internet/Abuse/Spam/Blacklist%s/`, 2007.

[8] INCORPORATED, G. Live-feed anti-phishing blacklist. `http://sb.google.com/safebrowsing/update?version=goog-black-url:1:1`, 2007.

[9] JUNG, J., PAXSON, V., BERGER, A. W., AND BALAKRISHNAN, H. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy 2004* (Oakland, CA, May 2004).

[10] KATTI, S., KRISHNAMURTHY, B., AND KATABI, D. Collaborating against common enemies. In *Proceedings of the ACM SIGCOMM/USENIX Internet Measurement Conference* (October 2005).

[11] KIM, H.-A., AND KARP, B. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium* (2004), pp. 271–286.

[12] LOCASTO, M., PAREKH, J., KEROMYTIS, A., AND STOLFO, S. Towards collaborative security and P2P intrusion detection. In *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security* (June 2005).

[13] M.GORI, AND PUCCI, A. Itemrank: A random-walk based scoring algorithm for recommender engines. In *Proceedings of the International Joint Conference on Artificial Intelligence* (January 2007).

[14] PORRAS, P., BRIESEMEISTER, L., SKINNER, K., LEVITT, K., ROWE, J., AND TING, Y. A hybrid quarantine defense. In *Proceedings of the 2004 ACM Workshop on Rapid Malcode (WORM)* (October 2004).

[15] RUOMING, P., YEGNESWARAN, V., BARFORD, P., PAXSON, V., AND PETERSON, L. Characteristics of internet background radiation. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference* (October 2004).

[16] THOMAS, R. Bogon dotted decimal list v3.9. `http://www.cymru.com/Documents/bogon-dd.hml`, October 2007.

[17] ULLRICH, J. DShield global worst offender list. `https://feeds.dshield.org/block.txt`.

[18] VIXIE, P., AND RAND, D. Mail abuse prevention system (MAPS). `http://www.mail-abuse.com`, 1997.

[19] WISSNER-GROSS, A. D. Preparation of topical readings lists from the link structure of Wikipedia. In *Proceedings of the IEEE International Conference on Advanced Learning Technology* (July 2006).

[20] YEGNESWARAN, V., BARFORD, P., AND ULLRICH, J. Internet intrusions: global characteristics and prevalence. In *Proceedings of ACM SIGMETRICS* (June 2003).

[21] YEGNESWARAN, V., PORRAS, P., SAIDI, H., SHARIF, M., AND NARAYANAN, A. The Cyber-TA compendium honeynet page. `http://www.cyber-ta.org/Honeynet`.

[22] ZHANG, J., PORRAS, P., AND ULLRICH, J. The DSHIELD highly predictive blacklisting service. `http://www.dshield.org/hpbinfo.html`.

[23] ZHANG, J., PORRAS, P., AND ULLRICH, J. A new service for increasing the effectiveness of network address blacklists. In *Proceedings of the 3rd Workshop of Steps to Reduce Unwanted Traffic on the Internet* (June 2007).

[24] ZHANG, J., PORRAS, P., AND ULLRICH, J. Gaussian process learning for cyber-attack early warning. *to appear in Proceedings of SIAM Conference on data mining* (2008).