Global Intrusion Detection in the DOMINO Overlay System

Anonymous Submission

Abstract

Sharing data between widely distributed intrusion detection systems offers the possibility of significant improvements in speed and accuracy over isolated systems. In this paper, we describe and evaluate DOMINO (Distributed Overlay for Monitoring InterNet Outbreaks); an architecture for a distributed intrusion detection system that fosters collaboration among heterogeneous nodes organized as an overlay network. The overlay design enables DOMINO to be heterogeneous, scalable, and robust to attacks and failures. An important component of DOMINO's design is the use of active sink nodes which respond to and measure connections on unused IP addresses. This enables efficient detection of attacks from spoofed IP sources, reduces false positives, enables attack classification and production of timely blacklists.

We evaluate the capabilities and performance of DOMINO using a large set of intrusion logs collected from over 1600 providers across the Internet. Our analysis demonstrates the significant marginal benefit obtained from distributed intrusion data sources coordinated through a system like DOMINO. We also evaluate how to configure DOMINO in order to maximize performance gains from the perspectives of blacklist length, blacklist freshness and IP proximity. We perform a retrospective analysis on the 2002 SQL-Snake and 2003 SQL-Slammer epidemics that highlights how information exchange through DOMINO would have reduced the reaction time and false-alarm rates during outbreaks. Finally, we provide preliminary results from our prototype active sink deployment that illustrates the limited variability in the sink traffic and the feasibility of efficient classification and discrimination of attack types.

1 Introduction

Internet intrusions and large scale attacks can have a catastrophic affect, including stolen or corrupted data, wide spread denial-of-service attacks, huge financial losses and even disruption of essential services. For example, it has been estimated that the CodeRed I virus infected more than $359,000$ hosts, resulting in financial losses of over $2 billion [17, 27]. Given their potentially profound impact, detecting network intrusions and attacks is an important goal.

However, protecting networks from nefarious intrusions and attacks remains challenging for a number of reasons. First, and perhaps the foremost, is the fact that the problem is a constantly moving target due to continued innovation, easy access to new portscanning tools and the Internet's basic vulnerability to widespread intrusions from different classes of worms [44]. Second, even when new exploits are identified, the primary means for propagating this information is through organizations, such as CERT [7], which can result in unacceptably slow response times for installing counter measures. Third, while infrastructures such as IP supported traceback [41] or pushback [26] offer promise in combating intrusions and attacks, these and other similar measures are not yet widely deployed.

Current best practice for protecting against intrusions is through the use of firewalls or network intrusion detection systems (NIDS) [31]. Firewalls are choke points that filter traffic at network gateways based on local security policies [5]. NIDS systems are monitors residing at end systems that passively observe the local network traffic and react to specific signatures (*misuse detection*) or statistical anomalies (*anomaly detection*). Examples of NIDS that employ *misuse detection* are Snort [39] and Bro [34]. One of the fundamental weaknesses of misuse-detection-based NIDS is their inability to detect new types of intrusions. *Anomaly detection* techniques establish statistical profiles of network traffic and flag any traffic deviating from the profile as anomalous. The high variability common in network packet traffic limits the effectiveness of this approach [23]. In general, current NIDS suffer from two major drawbacks: high false alarm rates and perspective from a single vantage point, which limits their ability to detect distributed or coordinated attacks.

One promising approach to addressing the above-mentioned shortcomings is through the use of *distributed network intrusion detection systems* (DNIDS). In this environment, alerts from different NIDS are combined to address above-mentioned shortcomings. Valdes and Skinner [48] show that "merging" alerts from different NIDSs deployed in a single administrative domain can reduce the overall false alarm rate. Improvements even from this limited perspective indicate the potential for DNIDS.

The first contribution of this paper is in the description of a new architecture for distributed intrusion information sharing. The DOMINO architecture enables NIDS deployed at diverse locations to securely share intrusion information. DOMINO's overlay design facilitates scalable data sharing, heterogeneous participation and robustness to nodes joining and leaving the infrastructure. DOMINO's data sharing architecture describes the methods of transfer and summarization of information between nodes. This architecture is flexible so as to enable consideration of local policies.

A important part of DOMINO's architecture are nodes that monitor *unused* IP addresses. We call the collection of these nodes the DOMINO *active sinks*. These data sources are devoid of false positives since they monitor unused IP addresses. The active sinks provides better mechanisms to detect spoofed sources and allows for efficient classification of attack packets into well defined categories. There is an important additional benefit in monitoring unused IP addresses in that there may be fewer privacy concerns associated with collecting this data.

The second contribution of this paper is in the evaluation of the DOMINO's design and performance characteristics. Our evaluation is based on the use of a set of intrusion logs gathered from over 1600 different networks across the Internet over a four month period. To our knowledge, this is the first evaluation of the DNIDS capability using a large, distributed dataset, and it provides key insights into effectiveness of distributed intrusion detection. We have completed an architectural specification of DOMINO and implementation is ongoing. Monitoring components have been developed while the implementation of message passing components are ongoing. Therefore, we performed an *ex post facto* analysis based on DOMINO's specification. We also evaluate data from a prototype active sink deployment. Our experiments focused on evaluating the following aspects of DOMINO:

- The marginal utility of adding measurement nodes in detecting worst offenders and creating port summaries.
- Ideal configuration parameters for the DOMINO architecture focused on blacklist size (a sorted list of the worst offending sources) and frequency of blacklist generation.
- The reaction time in identifying worm outbreaks.
- The effect on false alarm rates.
- The variability in payload distributions in active sink data.

There are several important results of our experimental investigation:

• **Improved summaries:** Through our marginal utility experiments we demonstrate that through a small network of collaborating peers (approximately 40), individual networks can significantly improve their perspective on global attack behavior. The size of the individual peering nodes is less significant than the number of collaborating peers.
• **Blacklists (Worst Offender List):** We show that few (approximately tens) attack sources are responsible for a significant portion of all scans on any given day and that substantial benefit can be achieved even through relatively stale blacklists.
• **Decreased reaction time:** We evaluate the reaction time of our system using data from two different outbreaks: SQL-Snake 2002 and SQL-Sapphire 2003 [30]. We provide examples of rules that DOMINO could employ to react favorably to each of these scenarios without significant false alarms. We demonstrate that reaction time to exploit recognition can be substantially reduced in DOMINO under each of these conditions.
• **Utility of active sink data:** We provide preliminary results from our active sink deployment that highlight the limited variability in observed payloads of malicious traffic and motivate our approach towards building a robust classifier.

Our results have a number of important implications. First, the DOMINO architecture demonstrates a framework within which systems from different administrative domains can participate in coordinated intrusion detection. Second, the clear improvements in ability to identify intrusions through coordinated data sharing should make this a compelling consideration for network administrators. Third, the deployment of active sink nodes on unused address space in the Internet would significantly increase the fidelity and speed of alert generation in intrusion detection systems. As a widely deployed infrastructure, DOMINO itself must be considered a target for attacks. To be effective, DOMINO must be resilient to a variety of attacks. A discussion of possible threats and corresponding mitigating measures in DOMINO are discussed in appendix E.

2 Related Work

There are several techniques for intrusion detection, such as misuse detection [34, 38], statistical anomaly detection [25, 43, 49], information retrieval [3], data mining [22], and inductive learning [46]. For a survey of intrusion detection reader can consult existing literature on this topic [1, 31, 33]. A classification of intrusion detection systems appears in [18, Section II].

Several researchers have started investigating distributed network intrusion detection [4, 10, 19]. Our general architecture for the DOMINO DNIDS is presented in Section 3. To our knowledge, with the exception of Indra [19], all other proposed DNIDS use a hierarchical structure. While Indra proposes a peer-to-peer approach to intrusion detection, its organization is completely ad-hoc and serves only as a rule dissemination mechanism. DOMINO's design uses a combination of peer-to-peer and hierarchical components providing significant advantages over a purely hierarchical architecture. These advantages include simplified information sharing, scalability and fault tolerance.

Currently, DOMINO uses a "flat tuple space" to express various alerts. Several researchers are developing languages to express alerts [11]. As these languages are standardized, we plan incorporate them into DOMINO. Merging alerts from various sources has also been studied by various researchers [9, 48]. The merging algorithm in DOMINO is influenced by our experimental results. We are also investigating algorithms from data fusion [14] for this purpose. The goal of intention recognition is to correlate alerts (possibly emerging from different sources) to infer the plan of the adversary [10, 21]. In the context of DOMINO we are not working on this problem. However, we plan to incorporate an existing intention recognition module into DOMINO. We are also investigating the use of structures to reason about sequences of attacks, such as privilege graphs [12] and attack graphs [42], for the purposes of intention recognition.

Our work is also influenced by empirical studies of intrusion and attack activity. Moore *et al.* [29] examined the prevalence of denial-of-service attacks using backscatter analysis. In [27], the authors analyze the details of the CodeRed worm outbreak and provide important perspective on the speed of worm propagation. In a follow-on work, Moore *et al.* [28] provide insights on the speed at which counter measures would have to be installed to inhibit the spread of worms like Code Red. The work that is perhaps most closely associated with ours is [50]. In that paper, the authors explore the statistical characteristics of Internet intrusion activity from a global perspective. That work informs DOMINO's design from the perspective of the potential use of multiple sites in coordinated intrusion detection.

3 DOMINO Architecture

3.1 DOMINO Overview

A DOMINO network is a dynamic infrastructure composed of a diverse collection of nodes located in a network spanning the Internet. The objective of this system is to provide a framework for information sharing aimed at improving intrusion detection capability for all participants. There are several overarching requirements, properties and challenges in organization of this network. These requirements are not unlike those of other large information sharing infrastructures and include the following:

- **Availability:** Since all networks are prone to system failures, congestion and attacks, the infrastructure must be resilient to temporary network instabilities. Furthermore, it is crucial that the network remain available in the face of *worm outbreaks*, *denial-of-service* attacks and other Internet catastrophes.
- **Scalability:** The success and utility of this network for its participants relies on its ability to scale gracefully to a large number of nodes.
- **Decentralization**: A decentralized architecture provides for greater flexibility and eliminates any single point of failure. The goal of this network is to advance the state of intrusion detection by enabling peer-to-peer collaboration between a large number of independent networks. In principle, however there could be instances of the DOMINO network that strictly operate locally inside an organization's Intranet.

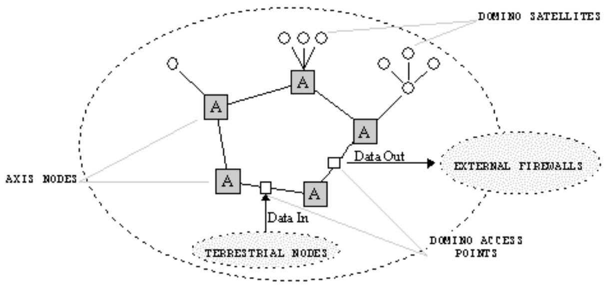


Figure 1: DOMINO Node Organization. Axis Nodes participate in a peer to peer overlay.

a)	Timestamp
b)	Protocol
c)	Sequence No
d)	Source IP
e)	Source Port
f)	Target IP
g)	Target Port
h)	TCP Flags
i)	Payload
j)	IP TTL
k)	Vulnerability ID
l)	Sensor ID (if data from a satellite node)
m)	Episode Type
n)	Episode ID

Table 1: Axis Database Schema (PacketLogs)

• **Pervasiveness**: The network would be most effective in identifying attack trends and characterizing global Internet intrusion phenomenon, if it obtains representative participants across a "moderate size" portion of IP address space.
• **Privacy**: The network should not reveal data that individual participants consider sensitive. It should also not increase the likelihood of attack against individual participants.
• **Heterogeneity**: The network must be able to harmonize systems from disparate networks of varying sizes that run a wide range of NIDS and firewall technologies. This would allow DOMINO to overcome any weaknesses associated with individual NIDS rules or organizational topologies.
• **Inducement**: Finally there must be an incentive (a direct benefit) for networks to join this infrastructure. The critical mass of participants required for obtaining immediate benefit should be reasonably low.

As shown in Figure 1, a DOMINO network is comprised of three sets of participants: *axis overlay, satellite communities and terrestrial contributers*. We describe each of these in the following sections. All communication between the axis overlay nodes and the satellites is encrypted. We provide a brief description of the key distribution strategy in Section 3.3.

3.1.1 Axis Overlay

The axis nodes are the central component of the DOMINO architecture. They are responsible for the bulk of the intrusion information sharing. Hence, their scalability and availability is vital to the resilience of the infrastructure. Two important requirements are 1) resilience to failure of axis nodes and 2) the ability to quickly detect and adapt to topological changes resulting from nodes joining and leaving.

Overlay networks have been shown to be highly resilient to disruption and possess the ability to deliver messages even during large-scale failures and network partitions [2]. We have chosen to organize the DOMINO axis nodes as a peer-to-peer overlay. We use the Chord lookup protocol [45] to facilitate the overlay. Chord has been demonstrated to provide the high availability and fault tolerance for peer-to-peer systems. However, the use of Chord itself is not essential. Extensions of this concept to other peer-to-peer key searching strategies like Pastry [13], or Tapestry [52] would be straightforward. The DOMINO network utilizes the Chord lookup protocol for two purposes: 1) to map IP ranges to axis nodes 2) to store and retrieve axis summaries.

In order to enhance robustness and extend the availability of the architecture, external connectivity (from nodes not participating in DOMINO) to the axis overlay is maintained through a set of *DOMINO access points (DAP)*. These nodes also participate in the Chord key distribution and lookup just like any other axis nodes. However, they do not perform any local monitoring or support Satellite communities which are primary functions of axis nodes. Participation at the axis node level in DOMINO is achieved through an administrative procedure, described in Section 3.3.

Each axis node in the overlay is described in terms of its following components:

Intrusion Data Collection: Axis nodes will act as intrusion data collection points in DOMINO. Axis nodes typically belong to large and well-managed networks since there is a high level of trust required to participate at this level. Each of these networks NIDS and/or firewalls and/or active sinks are deployed.

NIDS/Firewall: NIDS and firewall logs provide data on specific intrusion signatures and on rejected packets. Both of these are fundamental intrusion data sources in DOMINO.

Active sinks: A sink hole is a large chunk of unused (but advertised) IP address space. An active sink sniffs traffic bound to these addresses addresses and simulates virtual machines by supporting some level of interaction. A naive instantiation of an active sink is a *LaBrea tarpit* [24]. LaBrea was developed as a mechanism for slowing CodeRed I propagation by creating a "sticky honey-pot" or persistent connections over sink-holes. These cause the infecting machines to temporarily get stuck thus slowing propagation of an outbreak. Other examples of active sinks include *honeyd* (a highly customizable low-interaction honeypot) and *isink* (a scalable sink hole responder) [36, 51].

This approach to monitoring has important auxiliary benefits to DOMINO that includes the following:

• Active sinks enable examination of the payload packets. This helps in associating an attack with a particular vulnerability. For example, examination of the "GET" request helps distinguish between CodeRed, Nimda and other variants. This is not possible in traditional NIDS unless you have a service running on that port.

• Spoofed sources behave differently to an active sink response. They do not send the payload packet. Instead, they respond with a reset or simply drop the SYN/ACK received from the sink. Thus any source that sends a payload to the active sink is guaranteed to be malicious or is a result of mis-configurations. This enables creation of high confidence blacklists and attachment of greater accountability to the attack sources.

To assess the feasibility and scalability of a large scale active sink deployment, we have been running a version of an active sink (analogous to a tarpit) on 3 class B networks over 4 weeks. The number of monitored IPs were increased from around 50,000 to 100,000 during the measurement period. Characterization and analysis of data captured at the sink is discussed in appendix D.

Each axis node ideally maintains both an NIDS and an active sink over large unused IP address space. Our experience with similar datasets, as discussed in Section 4, indicates that a collection of around 20 such data sources is sufficient to identify global attack characteristics with a high degree of accuracy. Hence, we expect the number of axis nodes to be consistently over 20 in order to maximize effectiveness of the system.

Axis Database: The schema of the axis database has five important relations: packet logs, local and global summaries, vulnerabilities and alerts. To simplify discussion we treat these as flat relations. However, a hierarchical representation, such as XML, might be more suitable for implementation. For every packet that is received a DOMINO axis node logs the information in Table 1. Fields k), l), m) and n) are updated periodically by an analysis daemon since they might require examination of multiple packets.

DOMINO Summary Exchange Protocol: The DOMINO axis nodes in the overlay participate in a periodic exchange of intrusion information. We refer to the data sets exchanged as *summaries* - the actual format of the summaries is described later in this section. The summaries are exchanged at three levels of granularities: *hourly, daily and monthly*. A summary exchange involves the following steps:

1. Pulling data from the satellites. Alternatively this could also be implemented as a periodic push. The choice is left to the satellites.

2. Generation of the summary data and multicast to other axis nodes.

3. Generating appropriate Chord identifier and executing the store operation to enable persistent availability of this data.

DOMINO Query Engine: The DOMINO axis nodes export a queriable interface that can be used to tune firewall parameters and to expeditiously react to outbreaks. Queries from external sources are directed through the DAPs and their accessibility is controlled to protect the integrity of the infrastructure. Finally, the query engine also supports a "trigger" mechanism that allows the axis nodes to *pull* data from the satellites on a real-time basis. Such mechanisms can prove extremely valuable for gathering fine-grained information for analyzing new outbreaks.

3.1.2 Satellite Communities

Satellite communities are smaller networks of satellite nodes that implement a local version of the DOMINO protocol. There is potentially a wide disparity in the sizes and underlying NIDS and firewall software running in these networks, and extensions to provide support for DOMINO would be implemented as plug-ins for these systems.

The satellite nodes are organized in a hierarchy such that each node routes all communication with the larger network through a parent node that is either another DOMINO satellite or an axis node. Data collected at the satellite nodes is transmitted to the axis nodes through a combination of push and pull mechanisms. The data obtained from satellites is considered to be less trustworthy than what is collected at the axis nodes.

The satellites have the potential to generate a large volume of spontaneous alerts. Due to their limited perspective, these nodes may also be incapable of performing local analysis or classification of attack severity. Hence, these nodes are organized into ad-hoc hierarchies that allows for efficient clustering of neighboring alerts and robust construction of pertinent digests. Preserving hierarchical attributes towards the edges of the DOMINO overlay also facilitates efficient data aggregation, intelligent routing of queries and responses, establishment of trust levels and simplifies administrative demands.

Axis nodes and satellites enjoy a symbiotic relationship. The representation of the satellites allows the network wider coverage across the IP space. The *inducement* for the satellites is a global vantage point that allows for rapid outbreak recognition, dynamic content filtering and application specific source blacklisting to protect their networks in a timely manner.

3.1.3 Terrestrial Contributers

The terrestrial contributers form the least trustworthy but potentially a very large source of data. These nodes do not implement the DOMINO protocol, may not have active sinks and are not bound to any particular software installation. Rather, these nodes could run any firewall or NIDS software and simply supply daily summaries of port scan data. Terrestrial contributers are simply a means for expanding coverage by including intrusion data sets from outside of the infrastructure.

3.1.4 DOMINO Messages

To foster interoperability and maximize extensibility the DOMINO protocol messages are represented in XML. We extend the schema proposed by the IDWG (Intrusion Detection Working Group) in the IDMEF (Intrusion Detection Message Exchange Format) draft [16]. Our schema adds five new message types to the two provided by the IDMEF (alerts and hearbeats). The seven message categories in DOMINO are as follows: 1) Alerts 2) Summary Messages 3) Heartbeats 4) Topology Messages 5) Queries 6) DB Updates and 7) Triggers. A description of each of these message types and their formats is given in Appendix A.

3.2 Information Sharing

Every axis node maintains a local and global view of intrusion and attack activity. The local view considers activity in its own network and its satellites. Axis nodes periodically receive summaries from their peers which are then used to create the view of global activity. Issues in creating these views include scalability, timeliness and trust. Each axis node can employ its own strategy for creating both local and global views. Strategies for data aggregation to create local and global views are discussed in appendix B.

3.3 Authentication

The axis nodes in DOMINO are associated with a high degree of trust so authenticating all inter-axis communication is vital. We currently use public-key cryptography (specifically RSA [37]) for this purpose. However, other schemes for source authentication could also be used. We do not anticipate the number of axis nodes to scale at the same rate as the overall DOMINO infrastructure, so key distribution among these nodes is not envisioned as a big hurdle. In fact, there could easily be a special certificate authority (CA) for the DOMINO network, and when a new axis node joins DOMINO, it can engage in a key distribution protocol with the DOMINO CA. Certificates for the axis nodes are stored and retrieved through the Chord lookup protocol, much like all other data.

When an axis node multicasts an intrusion summary, it first computes a SHA-1 hash of the summary and appends the digital signature of the hash to the summary which is verified by all recipients. This approach is scalable in DOMINO because axis nodes broadcast summaries relatively infrequently and the summaries are lightweight (order of KBs). For example, in our current implementation the broadcasting period is approximately one hour. However, we plan to undertake an experimental evaluation of the overhead of computing digital signatures in the context of DOMINO. We are also investigating other mechanisms for source authentication (eg. [35, 40]), including elliptic-curve based public-key systems [20]. The public-key of an axis node can also be used for authentication using a standard challenge-response protocol (for example, see [32]).

Finally, authentication schemes based on secret key exchanges could also be considered. We chose not to pursue an authentication scheme based on sharing secret keys, since this would entail sharing a secret key between every pair of axis nodes. This approach would be less scalable and require more maintenance than our choice of using a public-key system.

4 Results

In this section, we first provide background results that demonstrate the utility of sharing intrusion information. In particular, we measure the amount of information that is gained by adding additional measurement nodes. We next investigate temporal attributes like the stability of blacklists, effectiveness of blacklist in terms of its size and the similarity of blacklists with respect to destination IP proximity. We also explore an how information sharing infrastructure would affect reaction times during a worm outbreak. The aforementioned results are all based on data obtained from DSHIELD [47]. We used a set of firewall and NIDS logs of portscans collected over a 4 month period from over 1600 firewall administrators distributed throughout the globe as the basis for analysis. Detailed description of the data appears in appendix C. Description and analysis of the data captured at the active sink is given in appendix D.

4.1 Marginal Utility

We use an *information theoretic* approach to quantify the additional information that is gained by adding new nodes in a distributed intrusion detection framework. Our approach utilizes the well known *Kullback-Leibler* [15] distance metric for probability distributions to measure the information gain.

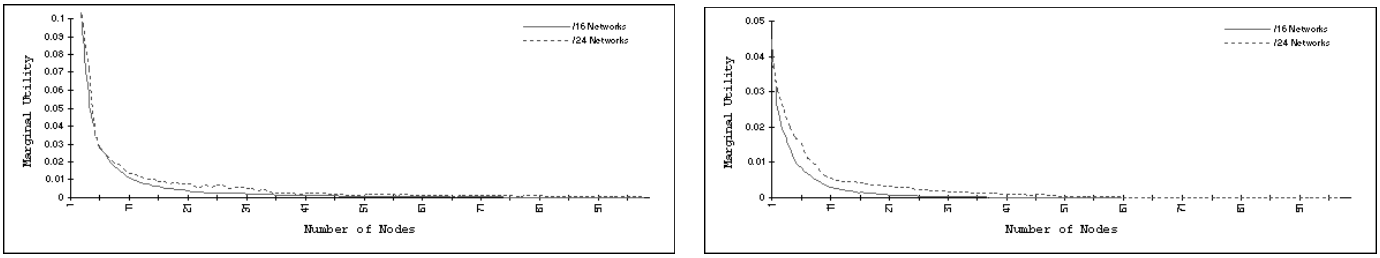
A framework for evaluating the marginal benefit of employing additional measurement sites in the context of Internet topology discovery has been presented in [6]. They presented two methodologies for quantifying the marginal benefit obtained by incorporating results from an additional experiment: *online* and *offline* marginal utility metric. The offline metric considers the benefit of each experiment on an *ex post facto* basis, measuring each experiment's usefulness after all the experiments have been conducted. In our study, each experiment corresponds to an additional intrusion log submitted from a different network and we choose the *offline* metric as we are not concerned with the order in which the logs are submitted.

Assume that we have n intrusion logs $S^1, \cdots, S^n$. Each log S^i defines a distribution P^i over the source ports that originate a scan, i.e., $P^i(s)$ is the probability that a scan originated from port s given the intrusion log S^i. We rank the intrusion logs by the entropy of the corresponding distribution, i.e., for $i < j$, P^i has higher entropy than P^j. Intuitively, a probability distribution with higher entropy contributes "more" to the overall distribution. Let $P^{[1,\cdots,i]}$ be the distribution when the information in the logs $S^1, \cdots, S^i$ are combined and let P be the overall distribution (when all the intrusion logs are combined). The marginal utility of S^i (denoted by $U(S^i)$) is:

$$
\begin{aligned}
U(S^i) &= d_{KL}(P^{[1,\cdots,i]}, P) \\
&= \sum_s P^{[1,\cdots,i]}(s) \log\left(\frac{P^{[1,\cdots,i]}(s)}{P(s)}\right)
\end{aligned}
$$

In the equation given above, the sum ranges over all the source ports that appear in the intrusion log.

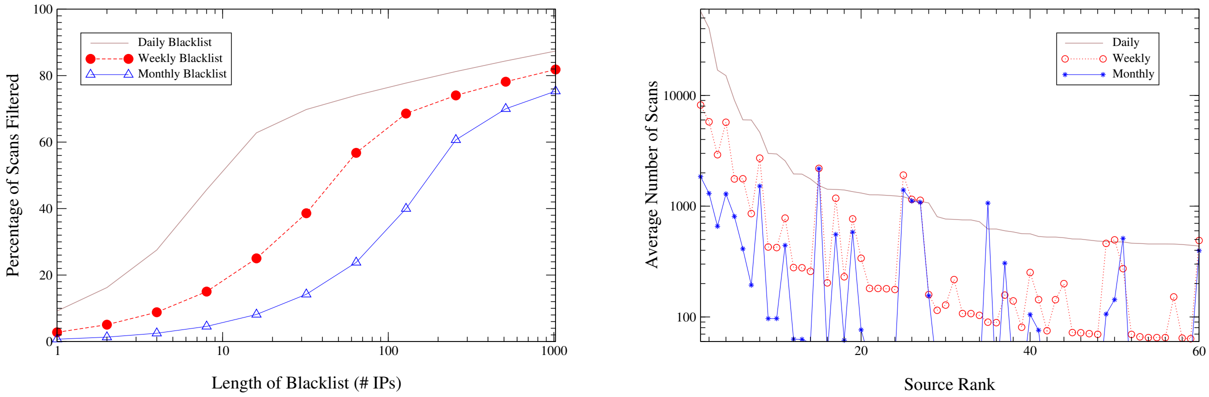
We use this framework to measure the effectiveness of sharing logs in identifying the worst offenders and the effectiveness of identifying the most frequently scanned target ports. For each day in the month of June, we randomly



(a) Utility of additional subnets for detecting worst offender.

(b) Utility of additional subnets for detecting top target port.

Figure 2: Utilities



(a) Effectiveness of Blacklists with length.

(b) Effectiveness of Blacklists with Age.

Figure 3: Effectiveness of Blacklists.

selected 100 /24's and 100 /16's from the DSHIELD logs to determine the number of participating networks that are required to obtain a stable distribution.

Figure 2(a) depicts the diminishing marginal benefit of adding additional network logs for developing port summaries. The curves for /16 and /24 networks show a very similar trend with the additional benefit declining to almost zero at 20 and 40 networks respectively. The message here is that there is some benefit to having a bigger measurement networks, but clearly it is more important to have measurements from multiple vantage points.

The graph of the marginal benefit for developing worst offender list (or blacklist) is given in Figure 2(b). The message is even more pronounced in this graph; clearly *size does not matter, but more is better!* Together the graphs imply that a collaboration of 40-60 networks is adequate to develop port summaries and blacklists with a high degree of confidence. It is also interesting to note that the actual marginal utility values for worst offenders is higher than that for port summaries. This suggests that it is more important to add additional sites for developing blacklists than it is for creating port summaries.

Summary: Marginal utility of information used to detect target and source ports (for port scans) is minimal after 40 nodes. This suggests that with respect to identifying target ports and the worst offenders for port scans, a DOMINO network with approximately 40 axis nodes will suffice.

4.2 Blacklist Effectiveness

One of the crucial operational parameters for the DOMINO overlay is the size of the blacklists that are exchanged between the participants. The DOMINO axis nodes develop and exchange *service specific* blacklists at multiple granularities.

To study this, we generated a combined blacklist for all the DSHIELD providers at three different granularities (daily, weekly and monthly). Figure 3(a) illustrates the relationship between the blacklist length and its effectiveness in terms of the percentage of all scans blocked.

The graph shows that at any given hour, around 90% of all scanning activity can be attributed to about 1024 source IPs. More surprisingly, a global daily blacklist of 16 sources, account for more than 60% of all scans. Similar benefits can be achieved by a stale (monthly) blacklist of around 250 sources.

Summary: Few sources are responsible for a large fraction of all scans and many sources persist. Therefore, the size of the blacklists in the DOMINO network does not have to be very large.

4.3 Blacklist Aging

Figure 3(b) provides another means to visualize the aging of blacklists. We again create blacklist of the top 60 sources at multiple granularities and graph the "average daily number of scans" generated by each rank. For the higher ranks (top 10), the hourly blacklists clearly deliver superior performance. However, for the lower ranks there are instances where the monthly blacklist performs as well or better than the daily blacklist. This validates the need for *maintaining blacklists at multiple granularities, and suggests that at lower granularities there is greater benefit to creating longer blacklists*.

4.4 IP Address Proximity

IP address proximity is an important consideration in the organization of the DOMINO topology. There are two conflicting issues that must be resolved in the allocation of satellites to axis nodes. First, to minimize false alarms and to effectively cluster related scans and attack episodes, it would be beneficial to organize nearby nodes (or networks) under the same hierarchy (since scanning and attack tools are often designed to sequentially traverse IP space). However, for every axis node to obtain a composite view of the attack activity, it would be ideal to have data from a diverse set of IP blocks. We would like to understand the appropriate granularity of aggregation that maximizes this tradeoff.

We randomly selected 100 /24 networks and measured the similarity in their monthly blacklists for June 2002. We defined the IP distance between two networks A.B.C and X.Y.Z as follows:

$$Dist = |A - X| * 256^2 + |B - Y| * 256 + |C - Z|$$

To express the similarity between blacklists of two networks, we needed a metric that provides greater weight for a match of higher rank. The asymmetric similarity of list B_2 to B_1 is denoted by *sim(*B_1, B_2*)*. The symmetric similarity between lists B_1 and B_2 (denoted by *SymSimilar(*B_1, B_2*)*) is the average of *sim(*B_1, B_2*)* and *sim(*B_2,B_1*)*. Formally, the similarity metrics are defined as follows (l denotes the length of the lists B_1 and B_2):

$$\begin{aligned} sim(B_1, B_2) &= \sum_{s_i \in B_1 \cap B_2} [l - rank(s_i, B_2)] \\ SymSimilar(B_1, B_2) &= \frac{sim(B_1, B_2) + sim(B_2, B_1)}{2} \end{aligned}$$

Figure 4 shows the similarity between blacklists as a function of the IP distance between two networks. The figure clearly shows that there is a high degree of similarity between the blacklists of /24 networks that are close together (in the same /16) and little similarity farther away.

Summary: Similarity of the two blacklists is positively correlated with the IP distance between their respective networks. This observation has several consequences in the context of DOMINO. First, satellite nodes in the same /16 IP address should be organized under a single axis node and that the set of /16 address spaces should be randomly distributed among the axis participants. Second, when an axis node generates its version of global summary, simple aggregation would be likely to work just as well as weighted merging.

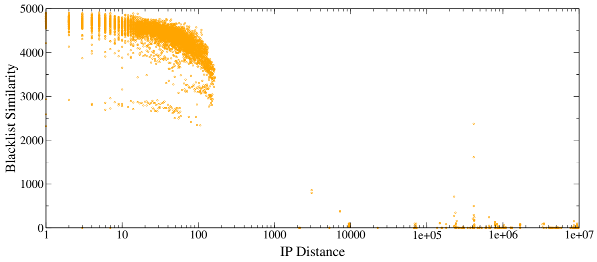


Figure 4: Similarity of Blacklists with IP proximity.

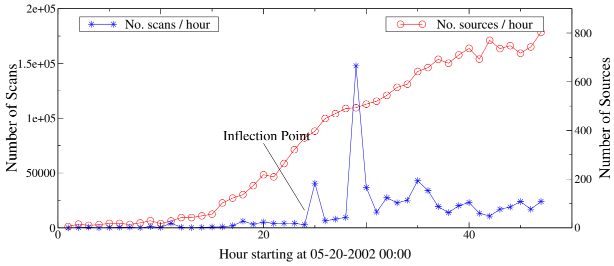


Figure 5: Scan Rate of 48 hrs surrounding SQL-Snake outbreak.

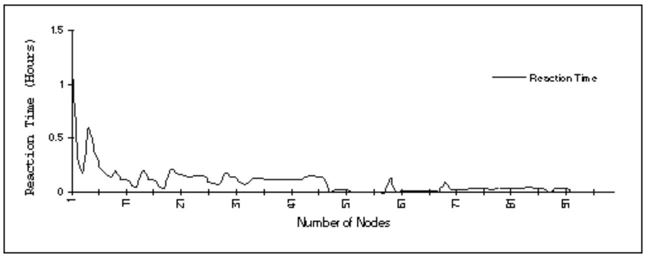


Figure 6: Reduction in Reaction Time as we add more networks

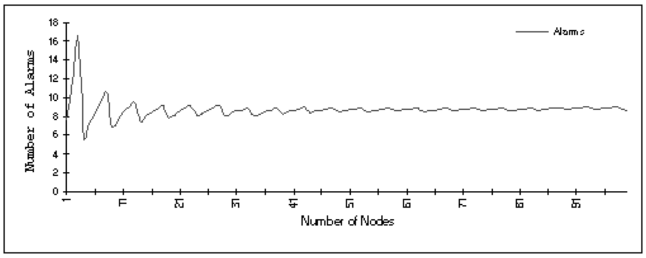


Figure 7: Change in Alarm Rate as we add more networks.

4.5 Retrospective Analysis: SQL Snake

In this section, we perform a retrospective analysis on the SQL-Snake Outbreak from May 2002. Unlike its precedents (CodeRed and Nimda) SQL-Snake was a relatively slow spreading worm, due to the small size of the susceptible population and its mode of propagation (TCP). We wanted to measure how information sharing through a system like DOMINO would affect *reaction time* and *alarm rate* during such an outbreak. We randomly selected 100 /24 networks and trained them with the port summary data of port 1433 (used by MS-SQL server) for the first two weeks of May. In particular, for each network we measured the hourly average number of scans and the average number of sources.

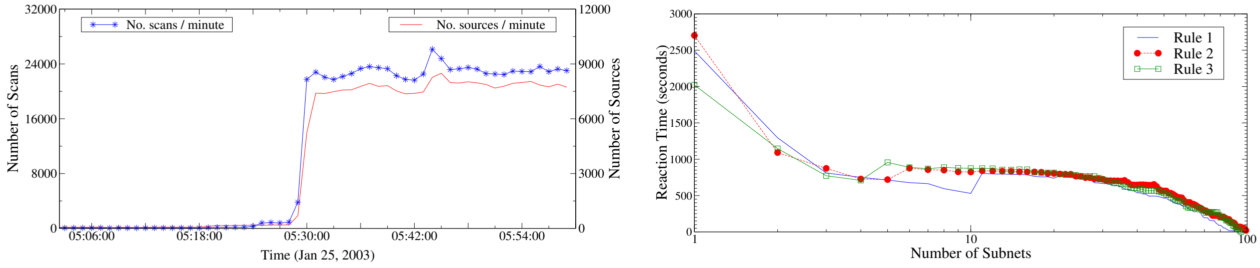
Figure 5 shows the hourly scanning rate in terms of the number of scans and the number of distinct sources scanning port 1433 during the 48 hours surrounding the outbreak. We denote the first visually apparent point of an outbreak (5/21, 00:00) as the *inflection point*.

We simulated 100 random iterations of DOMINO networks of axis nodes and in each iteration we measured the number of *outbreak alarms* generated in networks of of size ranging from 1 to 100 nodes. We assumed that the DOMINO nodes exchange hourly summaries of scanning activity but did not have any triggers that fired appropriate spontaneous alerts. In this experiment, we used a voting scheme to generate an *outbreak alarm*, i.e., an outbreak alarm is generated if atleast 20% of the nodes vote for an alarm. A node votes for an alarm if the following holds:[1]

- 200% increase in number scans from the hourly average, and
- 100% increase in the number of sources from hourly average, and
- number of sources is greater than five.

The *reaction time* is defined as the elapsed time between the *inflection point* and the first alarm after that point. Figure 6 shows the *decrease in observed reaction time* from an average of more than an hour with a single node to almost zero as we add sufficient axis nodes (approximately 50). Figure 7 displays the average number of alarms, which decreases with topology size and stabilizes at about 8. These alarms are not false alarms, but correspond

[1]We could have chosen a more complicated rule for generating alarms (for example, one based on statistical anomaly detection). However, this simple rule suffices to illustrate our point.



(a) Scan rates on port 1434 surrounding SQL-Sapphire Outbreak (01/25/2003 5:00-6:00am).

(b) Reaction Time for SQL-Sapphire Outbreak.

Figure 8: Statistics for SQL-sapphire..

exactly to the 8 preceding hours before the *inflection point* that show a gradual increase in the source rates and are points when the outbreak could have been predicted earlier by DOMINO. The oscillatory behavior of the alarm rate is an artifact of the rule that requires at least an integral 20% of the participants to vote for an outbreak.
Summary: By adding sufficient nodes, outbreaks can be detected early with minimal reaction time and few false alarms.

4.6 Retrospective Analysis: SQL-Sapphire

The SQL-Sapphire worm also known as SQL-Slammer was released in January 2003, and wreaked significant havoc on the networking infrastructures in under ten minutes. The worm distinguished itself from its predecessors by its small payload size (single UDP packet of 404 bytes) that enabled a rapid propagation rate in spite of a small susceptible population (75000) [30]. The reality of such high speed worms [44] implies that distributed architectures, such as DOMINO, might have the best opportunity to detect and react to such worm outbreaks.

Figure 8(a) shows the exponential increase in the number of scans and number of sources in the minutes following the outbreak. For such epidemics, alarms generated through hourly axis summaries do not suffice. DOMINO's mechanism to deal with such scenarios are *spontaneous alerts* that are issued through *triggers*.

Whenever possible, DOMINO nodes associate related packets with *episodes*, e.g., horizontal scan episode (sequential scan of several machines in the same subnet aimed at the same target port), vertical scan episode (scan of multiple ports of single IP to survey several vulnerabilities), and a coordinated scan episode (distributed scan of a subnet through multiple sources). For episodes on every port, DOMINO nodes maintain the average number of scans, the average number of attack sources and the duration. A trigger for a *spontaneous alert* can be defined as an episode that deviates from the average as follows:[2]

- number of sources is > 5, and
- the number of scans is > 10 times the average, or
- the number of sources is > 10 times the average, or
- the duration is > 10 times the average.

We recognize the existence of an outbreak when at least 10% (rule 1), 20% (rule 2) or 30%(rule 3) of the participants generate a spontaneous alert in the last hour. We repeated the previous experiment with 100 random iterations. In each iteration, we picked 100 random class-C subnets and used the data from first 2 weeks of January to train the system. We measured episode rates, simulated spontaneous alerts and then cataloged the change reaction time as we add additional subnets under each of the 3 rules. Figure 8(b) shows that by adding sufficient nodes,

[2] As in the previous subsection, we can use a more sophisticated rule to generate a spontaneous alert. However, a simple rule will suffice to illustrate our point.

the reaction time can be reduced to a few seconds. The goal of DOMINO is not *outbreak containment* but rather *outbreak recognition* and insulation of maximal number of participants [28].

5 Conclusions and Future Work

In this paper, we describe and evaluate DOMINO, a cooperative intrusion detection system. DOMINO is designed to enable intrusion information sharing in a globally distributed network consisting of: 1) trusted axis nodes organized in a peer-to-peer overlay, 2) satellite nodes associated with each axis node that are hierarchically arranged, 3) terrestrial nodes, which are deployed at the leaves of the infrastructure, that provide daily intrusion summaries. DOMINO's design is based on heterogeneous data collection through NIDS, firewalls and active sinks. This architecture enables DOMINO to be secure, scalable, fault-tolerant, and facilitates data sharing.

Our evaluation of DOMINO is based on data from two sources. The first is a set of intrusion logs collected over a four month period from over 1600 networks world wide. The second is from a prototype sink implementation on a single network which monitors over 100K IP addresses. Our evaluation clearly demonstrates the utility of sharing information between multiple nodes in a cooperative infrastructure. We use an information theoretic approach to show that perspective on intrusions can be greatly enhanced by cooperation of a relatively small number of nodes. Using the 2002 and 2003 SQL-worm outbreaks, we demonstrate that false-alarm rates can be significantly reduced in DOMINO and that reaction time for outbreak detection can be similarly reduced. Finally, we provide an initial evaluation of the effectiveness of active sinks in discriminating between types of attacks based on examining payload data. Our results clearly demonstrate that active sinks provide important insight in this regard. Based on these analysis, we conclude that DOMINO offers a significant opportunity to improve intrusion and outbreak detection capability in the Internet.

We intend to pursue future work in a number of directions. First, we plan to develop more interactive sink-hole responders that would further enhance attack discrimination capability. Secondly we plan to deploy an operational DOMINO infrastructure. This will enable us to test and further develop the DOMINO topology creation and maintenance protocols. As we expand, the infrastructure will enable case studies of future intrusion and outbreak activity. We also plan to investigate alternative methods for information merging and sharing with the goal of improving efficiency and precision. Finally, we plan to develop tools for automating firewall rule generation.

References

[1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner. State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99-TR-028, Software Engineering Institute, Carnegie Mellon, January 2000.

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of 18th SOSP*, Lake Louise, Alberta, October 2001.

[3] R. Anderson and A. Khattak. The use of information retrieval techniques for intrusion detection. In *Proceedings of First International Workshop on the Recent Advances in Intrusion Detection (RAID)*, September 1998.

[4] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, Eugene H. Spafford, and Diego Zamboni. An architecture for intrusion detection using autonomous agents. In *14th Annual Computer Security Applications Conference (ACSAC'98)*, pages 13–24, December 7–11 1998.

[5] S. Bellovin and W. Cheswick. Network firewalls. *IEEE Communications Magazine*, September 1994.

[6] P. Barford A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of network topology measurements. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.

[7] CERT Coordination Center. *http://www.cert.org*, 2001.

[8] CERT. Cert Incident Note IN-2002-06. *http://www.cert.org/incident_note/IN-2002-06.html*, 2002.

[9] Frederic Cuppens. Managing alerts in a multi-intrusion detection environment. In *Proceedings of the 17-th Annual Computer Security Applications Conference (ACSAC)*, 2001.

[10] Frederic Cuppens and Alexandre Miege. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of IEEE Symposium on Security and Privacy (Oakland 2002)*, pages 202–215, 2002.

[11] Frederic Cuppens and Rodolphe Ortalo. LAMBDA: A language to model a database for detection of attacks. In *Proceedings of Recent Advances in Intrusion Detection (RAID 2000)*, pages 197–216, 2000.

[12] M. Dacier. *Towards Quantitative Evaluation of Computer Security*. PhD thesis, Institut National Polytechnique de Toulouse, December 1994.

[13] P. Druschel and A. Rowstron. Storage management and caching in PASTRY, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of ACM SOSP '01*, Lake Louise, Alberta, October 2001.

[14] I.R. Goodman, R.P.S. Mahler, and H.T. Nguyen. *Mathematics of data fusion*. Kluwer Academic Publishers Group, Dordrecht, 1997.

[15] R. Gray. *Entropy and Information Theory*. Springer-Verlag, 1990.

[16] Intrusion Detection Working Group. *http://www.ietf.org/html.charters/idwg−charter.html*, 2003.

[17] IDG. Study: Code Red costs top $2 billion. *http://www.thestandard.com*, August 2001.

[18] K. Ilgun, R.A. Kemmerer, and P.A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[19] R. Janakiraman, M. Waldvogel, and Q. Zhang. Indra: A peer-to-peer approach to network intrusion detection and prevention. *Unpublished Manuscript*, 2003.

[20] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[21] Christopher Krugel, Thomas Toth, and Clemens Kerer. Decentralized event correlation for intrusion detection. In *Proceedings of Information Security and Cryptology - ICISC 2001, Seoul, Korea*, pages 114–131, December 6–7 2001.

[22] W. Lee, S.J. Stolfo, and K.W. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, 1999.

[23] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, pages 2:1–15, 1994.

[24] Tom Liston. Labrea homepage. *http://www.hackbusters.net* , 2003.

[25] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. garvey. A real-time intrusion detection expert system (IDES)-final technical report. Technical report, Computer Science Laboratory, SRI international, Menlo Park, California, February 1992.

[26] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. In *Computer Communications Review V32, N3*, July 2002.

[27] D. Moore, C. Shannon, and K. Claffy. Code red: A case study on the spread and victims of an internet worm. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[28] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM*, April 2003.

[29] D. Moore, G. Voelker, and S. Savage. Inferring internet denial of service activity. In *Proceedings of the 2001 USENIX Security Symposium*, Washington D.C., August 2001.

[30] David Moore, Vern Paxson, Stefan Savage, Collen Shannon, Stuart Staniford, and Nicholas Weaver. The Spread of the Sapphire/Slammer Worm. *http://www.caida.org/outreach/-papers/2003/sapphire/sapphire.html*, 2003.

[31] B. Mukherjee, L. Todd Heberlein, and K.N. Levitt. Network intrusion detection. *IEEE Network*, May/June 1994.

[32] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:993–999, 1978.

[33] S. Northcutt. *Network Intrusion Detection: An Analyst's Handbook*. New Riders, 1999.

[34] Vern Paxson. BRO: A System for Detecting Network Intruders in Real Time. In *Proceedings of the 7th USENIX Security Symposium*, 1998.

[35] A. Perrig. The BiBA one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, November 5-8 2001.

[36] Niels Provos. Honeyd - a virtual honeypot daemon. In *Proceedings of the 10th DFN-CERT Workshop*, February 2003.

[37] R.L. Rivest, A. Shamir, and L.M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Coommunications of the ACM*, 21:120–126, 1978.

[38] M. Roesch. Snort- lightweight intrusion detection for networks. In *Proceedings of the 1999 USENIX LISA conference*, November 1999.

[39] Marty Roesch. The SNORT Network Intrusion Detection System. *http://www.snort.org*, 2002.

[40] P. Rohtagi. A compact and fast hybrid signature scheme for multicast packet. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS)*, pages 93–100, 1999.

[41] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical Network Support for IP Tracback. In *Proceedings of ACM SIGCOMM 2000*, 2000.

[42] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of IEEE Symposium on Security and Privacy*, May 2002.

[43] S. Staniford, J. Hoagland, and J. McAlerney. Practical automated detection of stealthy portscans. In *Proceedings of the ACM CCS IDS Workshop*, November 2000.

[44] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[45] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM '01*, San Diego, CA, August 2001.

[46] H.S. Teng, K. Chen, and S. C-Y Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *IEEE Symposium on Security and Privacy*, pages 278–284, 1999.

[47] Johannes Ullrich. DSHIELD. *http://www.dshield.org*, 2000.

[48] Alfonso Valdes and Keith Skinner. Probabilistic alert correlation. In *Proceedings of Recent Advances in Intrusion Detection (RAID 2001)*, pages 54–68, 2001.

[49] C. Warrender, S. Forrest, and B. Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.

[50] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *Proceedings of ACM SIGMETRICS*, June 2003.

[51] Vinod Yegneswaran, Paul Barford, and Dave Plonka. On the Design and Utility of Internet Sinks for Network Abuse Monitoring. $UW Technical Report$, 2003.

[52] B. Zhao, A. Joseph, and J. Kubiatowicz. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Univeristy of California, Berkeley, Computer Science Department, April 2001.

A DOMINO Messages

To foster interoperability and maximize extensibility the DOMINO protocol messages are represented in XML. We extend the schema proposed by the IDWG (Intrusion Detection Working Group) in IDMEF (Intrusion Detection Message Exchange Format) draft [16]. Our schema adds five new message types to the two provided by the IDMEF (alerts and hearbeats). The seven message categories in DOMINO are as follows:

Alerts - Alerts are spontaneous responses to events as defined by NIDS and firewall or custom policies. Most alerts are generated at the small networks or satellites, however they might get propagated to the axis level depending on the pervasiveness and severity. Alert clustering and suppression is a very challenging problem and vital to the operational success of the infrastructure. The IDMEF draft defines a few alert classifications: tool alert, correlation alert and overflow alert. The DOMINO axis nodes also exchange alerts when there is a significant deviation from the periodic summaries. For example, outbreak alerts, blacklist alerts and denial-of-service attack alerts. The DTD for an alert is as follows:

```
<!ELEMENT Alert (CreateTime, DetectTime?,
  AnalyzerTime?, Classification, Source*, Target*,
  AdditionalData?)>
<!ATTLIST Alert version CDATA #FIXED '1', ident
  CDATA #REQUIRED, impact CDATA 'unknown'>
```

Summary Messages - DOMINO summaries are typically exchanged by the axis peers in one of three possible formats relating to the type of information being transmitted. The summary message types include: Port Summaries, Source Summaries and Cluster Summaries. DOMINO also defines three levels of trust (low, medium and high) for summary messages based on their source (axis/satellite). The choice of three levels of trust is somewhat arbitrary and are used as cues for intelligent aggregation. The DTD for summary messages is as follows:

```
<!ELEMENT Summary (CreateTime, SummaryDuration,
  IPBlockSummary+)>
<!ATTLIST Summary version CDATA #FIXED '1', ident
  CDATA #REQUIRED>
<!ELEMENT IPBlockSummary (MinIP, MaxIP, IPCount,
  TrustLevel, PortSummary?, SourceSummary?,
  ClusterSummary?)>
<!ELEMENT PortSummary (VulnID/PortNum, NumUniqSrcIP,
  NumUniqDestIP, ScanCount>)
<!ELEMENT SourceSummary (VulnID/PortRange, ScanCount,
  AggregateScanCount, NumUniqTargets)>
<!ELEMENT ClusterSummary(SrcIPList, DestIPList,
  VulnID/PortRange, ScanCount)>
```

Heartbeats - In DOMINO the Satellite Nodes periodically exchange heartbeat messages with the parent nodes. These are used to indicate the current status to higher level nodes and vice-versa. These interval of heartbeats is left up to the satellites, it could be say every 10 minutes or every hour.

```
<!ELEMENT Heartbeat (CreateTime, AnalyzerTime,
  AdditionalData*)>
<!ATTLIST Heartbeat ident CDATA #REQUIRED>
```

Topology Messages - There are four different types of topology messages: *adopt, detour, recall, and divorce*. When a satellite node is disconnected from its parent, it tries to reconnect through the normal heartbeat exchange protocol. If this fails, it issues an *adopt* message to a DAP that is then multicast to the overlay of axis nodes. An axis node might forward the adopt message to any applicable children. The satellite analyzes the acknowledgments and responds with a *detour* message to the most eligible parent. When an axis or satellite parent restarts, it issues a *recall* message to all its children. The child can accept the invitation to rejoin by issuing a *divorce* message to the foster parent and a simultaneous *detour* message to the original parent.

```
<!ELEMENT TopologyMessage (CreateTime, Type,
  IPBlockSummary?>)
<!ATTLIST TopologyMessage version CDATA #FIXED
 '1', ident CDATA #REQUIRED>}
```

Queries - The DOMINO Query Messages are exchanged in XQuery format. Since the axis nodes maintain a consistent schema inter-axis queries could be done in SQL. However, we chose to use XQuery to maximize interoperability with satellites. We provide an example query which is to create a top 10 blacklist for port 1433 between two specified times:

```
for $src in distinct(document("localscans.xml"))//source
let $scan := document("localscans.xml")//:scan[source = $src]
let $time := $scan/timestamp, $port = $scan/port
  where $port = 1433 and $time > 1044206900
        and $time < 1044206960
  return
    <blacklist>
      <source> {$src} </source>
      <num_scans> {sum{$scan/count}} </numscans>
    </blacklist>
  } sortby {sum{$scan/count}} limit 10
```

DB Updates - The DOMINO protocol also provides an automatic mechanism for updating NIDS rulesets and the axis vulnerability database. This can also be considered as a means for dispensing timely content based filters to the satellites. The format of these messages is straightforward.

```
<!ELEMENT DBUpdate (CreateTime, VulnerabilityID,
  Signature)>
<!ATTLIST DBUpdate version CDATA #FIXED '1', ident
  CDATA #REQUIRED, description CDATA>
<!ELEMENT Signature (TargetPorts+, Payload?,
  SourcePort*, Protocol+, Seqno?)>
```

Triggers - Triggers can be issued by DOMINO axis and Satellites to nodes that are lower in the hierarchy. A trigger has three components 1) Query 2) Constraint and 3) Action. We define two types of actions: alerts and filter rules. An example of an trigger is the generation of an *outbreak alert* when the number of scans exceeds a certain threshold.

```
<!ELEMENT Trigger (CreateTime, Query, Constraint, Action)>
<!ATTLIST Trigger version CDATA #FIXED '1', ident
  CDATA #REQUIRED, description CDATA>
<!ELEMENT Action (Alert?, Filter?)>
```

B Information Sharing

Potential strategies for information sharing include the following:

Local aggregation: Once intrusion information has been gathered at the satellite nodes, the next step is to consider how to organize and refine the data to create a coherent picture of malicious activities. Moreover, if satellite nodes send "raw alerts", the axis nodes will get overwhelmed. Therefore, alerts from the satellite nodes need to be "aggregated" before they are communicated to the axis nodes. Cuppens [11] describes a *cooperative intrusion detection module* or *CRIM* as a means for combining alerts from different IDSs. DOMINO adopts and extends this design for the purposes of aggregating "raw alerts".

Global aggregation: The most straight-forward way to merge logs from multiple sites is through a simple addition or average across each dimension of data. While this approach provides a simple means for organizing and summarizing data, it also has the risk of inaccuracy. As an example, consider the case of a Port Summary. It seems logical to add the the number of scans and the number of unique destinations, but simply adding the set of *unique sources* across axis nodes is almost certainly not appropriate. DOMINO currently performs simple aggregation for Port Summaries (but does not consider the results for sources).

Weighted merging: A potentially important consideration in fusing summaries is IP address proximity. In particular, summaries generated from "neighboring" IP address blocks might be more germane than those generated in a "distant" network (since it is not uncommon for scans or attacks to proceed horizontally through IP space). A weighted merging approach that emphasizes proximity might be more appropriate. DOMINO currently performs a

Table 2: Monthly Summary of studied DSHIELD Logs

Month	No: Scans	No: Dest IPs
May. 2002	48 million	375,323
June. 2002	61 million	382,224
July. 2002	68 million	402,050

Table 3: Sample log entries from Portscan logs

Date	Time	Sub. Hash	No: Scans	Src IP	Src Port	Targt IP	Targt Port	TCP Flags
2002-03-19	18:35:18	provider2323	3	211.10.7.73	1227	10.3.23.12	21	S
2002-03-19	18:35:19	provider2323	16	211.10.7.73	1327	10.3.23.12	53	SF

very simple weighted merging of blacklists.

Sampling: Sampling is the standard method for reducing the scale of measurement data. The goal in any sampling approach is to balance quantity of data with precision of measurement. In the case of DOMINO, this is challenging since intrusions can take the form of attacks (which would be easy to sample) and stealthy scans (rare events which are hard to sample). Any sampling method used in DOMINO would have to poses the ability to expose both types of events. We are investigating the feasibility of employing sampling as a technique for data sharing.

A related issue that is important in DOMINO is the aging of local data. The packet data accumulated in large sinks could be on the order of 100's of Megabytes per day. Summaries, however, are meant to be light weight so simply purging data older than a certain number of days might be a reasonable approach in practice. However, care must be taken to ensure that periodic patterns, such as the monthly rise and fall of the CodeRed worm, are not lost. At present, we propose that DOMINO maintain summaries at several granularities and uses weighted averaging to merge older summaries with more timely data.

C Intrusion Trace Data

We use a set of firewall and NIDS logs of portscans collected over a 4 month period from over 1600 firewall administrators distributed throughout the globe as the basis for analysis of DOMINO. The logs provide a condensed summary (lowest common denominator) of portscan activity obtained from various firewall and NIDS platforms. Some of the platforms supported include BlackIce Defender, CISCO PIX, ZoneAlarm, Linux IPchains, Portsentry and Snort. This approach significantly increases the coverage and reduces reliance on individual NIDS's interpretation of events.

Table 3 illustrates the format of a typical log entry. The date and time fields are standardized to GMT and the provider hash allows for aggregation of destination IP addresses that belong to the same administrative network. Table 2 provides a high level summary of the data that was used in this analysis[3]. The dataset was obtained from DSHIELD.ORG – a research effort funded by the SANS Institute as part of their Internet Storm Center [47]. The goals of DSHIELD include detection and analysis of new worms and vulnerabilities, notification to ISPs of exploited systems, publishing blacklists of worst offenders and feedback to submitters to improve firewall rulesets. The data is comprised of logs submitted by a diverse set of networks and includes 5 Class B networks, over 45 Class C sized networks and a large number of smaller sub-networks. The networks represented in this data set are widely distributed both geographically and topologically in the autonomous system space. This provides a unique perspective on global intrusion activity highlighted by DSHIELD's contribution in the detection and early analysis of CodeRed, Nimda and SQL worm(s) outbreaks.

D Preliminary Results from the Active Sink

Figures 9 and 10 show the number of packets and flows per second respectively that were inbound and outbound from the active sink. The positive flows are outbound and the negative are inbound. As might be expected, the number of inbound packets is higher than outbound because the active sink does not respond to the persistent payload packets. The difference in the number of inbound and outbound flows is an artifact of the way flows are accounted over 5

[3]We also used DSHIELD data for port 1433 from January, 2003 for our SQL-Sapphire analysis.

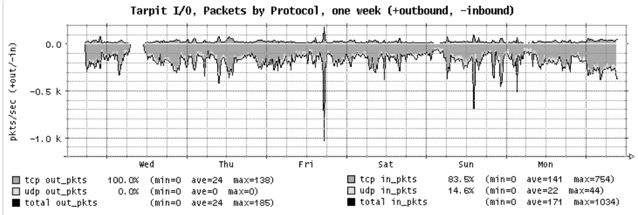


Figure 9: Protocol breakdown of active sink packets Jan 28 - Feb 4.

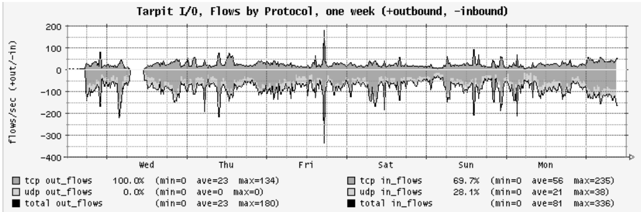


Figure 10: Protocol breakdown of active sink flows Jan 28 - Feb 4.

minute intervals. It should not be surprising that there are no outbound UDP packets. The number of inbound packets was typically between 200-300 packets or about 40-50 connection attempts per second. The active sink running on a Pentium 4 Linux PC, had no problem coping with this traffic rate since no per-connection state is maintained. The connection attempts spanned a wide variety of ports and originated from hundreds of thousands of sources. A typical summary of the top ports for a given week is shown in Table 4. The *ms-sql-s* and *ms-sql-m* scans correspond to the recent SQL-Sapphire worm and SQL-Snake respectively. The HTTP probes are from CodeRed and Nimda infected hosts. The microsoft-ds scans, port 139, port 135 scans are from the Lioten worm [8]. These are followed by scans for four different open proxy servers (often used as a means obfuscate Internet activity).

Table 4: Sample Weekly Summary Top Probed Services

Service	**Port**	**Protocol**	**Flows**	**Octets**	**Packets**
ms-sql-s	1434	UDP	548838	388453676	1371925
microsoft-ds	445	TCP	541528	42580046	545867
ms-sql-m	1433	TCP	301428	115385725	997172
http	80	TCP	249569	66851055	728766
netbios-ss	139	TCP	99075	10894702	230539
AnalogX (Proxy Server)	6588	TCP	82707	8594185	134813
https	443	TCP	69025	7988260	158725
HyView Proxy	3128	TCP	27483	1146324	27970
http-alt	8080	TCP	27109	1109656	27374
Win NT/2000 RPC	135	TCP	6765	291224	7279

An important application of the traffic captured by the active sink nodes is generating signatures for malicious payloads, e.g., signature for a payload of a worm. Currently, NIDS use simple pattern matching to identify malicious payloads. This method can lead to significant number of false positives because variations in malicious payloads cannot be detected. We demonstrate how the traffic captured by the active sink nodes can be used to create a more "robust" signature for a malicious payload.

Our first step is to cluster the payloads of the traffic observed at the active sink nodes. Intuitively, each cluster corresponds to malicious payload. Next, we construct a classifier for each cluster. These classifiers can then be used by a NIDS to identify malicious payloads. We have only performed the clustering step. In the future, we will investigate constructing classifiers and their use in identifying malicious payloads. However, the results of the clustering are encouraging.

We performed clustering on data collected between Jan 6, 2003 and Jan 28, 2003. First, we constructed a *fingerprint* for each payload. A fingerprint for a payload is the distribution over bytes between `0x1F` and `0x7E` (these are the same bytes that are used by Snort in displaying payloads). Each fingerprint also records the number of bytes that were outside this range. The distance between two payloads is the Kullback-Leibler distance between their fingerprints. Payloads were clustered using the *k-means* [15] algorithm and the sum of squared metric was used to determine the optimal number of clusters.

Our results show that there are six distinct clusters (see Table 5). Figure 11 provides a cumulative distribution function of the distance from the cluster centers. Clusters 1 and 3 are perfect clusters (distance of zero). The clusters

Cluster	Port (No. Scans)
cluster1	445 (1090338)
cluster2	80 (1315982), 3128 (10995), 8080 (24066)
cluster3	139 (160668), 443 (27377), 3128 (7181)
cluster4	135 (5791)
cluster5	23 (29108), 80 (2309958), 8080 (10770)
cluster6	1433 (2167842)

Table 5: Cluster Summary

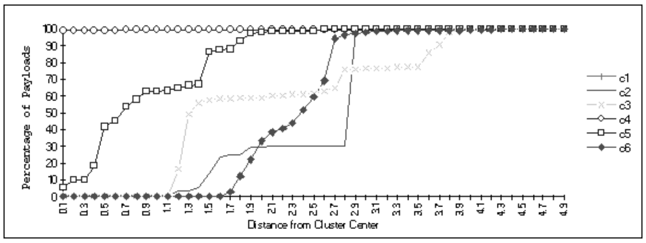


Figure 11: Variability in the payload clusters.

with port 80 (2 and 5) and port 1433 seem to have little more variability. Port 8080 scans in cluster 2 and 5 appear to be CodeRed/Nimda variants. The variability in these clusters can be attributed to two reasons: each attack of CodeRed,Nimda and SQL-Snake is a series of similar packets that attempt to open a shell and execute a series of commands. There are several variants of these worms (especially true of port 80) that try a slightly different search path from the default for the presence of an exploit. Therefore, our experiments demonstrate that clusters naturally correspond to classes of malicious payload, so classifiers generated from these clusters should be successful in identifying malicious payloads.

E Threat Vulnerability

As a widely deployed infrastructure, DOMINO itself must be considered a target for attacks. To be effective, DOMINO must be resilient to a variety of attacks. While its design is robust, we have not attempted to address *all possible* vulnerabilities of DOMINO to attack. By virtue of the fact that its architecture enables heterogeneous client participation, it may well be infeasible to address all possible vulnerabilities. We address threats to DOMINO through a model that considers the *most likely* forms of attacks that may be attempted. These include attacks intent upon denying service in the infrastructure, attempts to infiltrate the infrastructure, and attacks intent upon reducing DOMINO's effectiveness.

E.1 Denial of Service

Threat: An attempt to effectively remove node(s) through DoS attack from systems outside of DOMINO.
Remedy: In the face of standard packet flood attacks, it is certainly possible that some set of DOMINO nodes could be effectively removed from the infrastructure. In fact, it is a non-goal of the infrastructure to protect nodes from DoS attack. However, the distributed, coordinated nature of the infrastructure makes it robust to the removal of nodes through failures or attacks.
Threat: A compromised DOMINO node begins sending large amounts of what appears to be legitimate data in an attempt to mount a DoS attack on another axis node.
Remedy: An axis nodes can apply filters to incoming data such that data sent by any node or set of nodes cannot exceed a specified threshold. The configuration of filters will be dependent both upon system resources and upon historical variability. If multiple axis nodes have been compromised, then filtering could cease to be effective.

E.2 Infiltration

Threat: An attempt to gain unauthorized access to an axis node.
Remedy: DOMINO is not specifically concerned with individual system security. We assume that standard best practices for hardening networked systems to intrusions, such as keeping up with operating system patches and closing all unused services, will be employed. Furthermore, we expect that the vulnerability of DOMINO specific software such as buffer-overrun exploits can be limited through best practices of software engineering.
Threat: An attempt to masquerade as an axis node.
Remedy: As discussed in Section 3.3 an axis node can be authenticated by other axis nodes. We assume that axis nodes are intermittently forced to participate in a mutual authentication protocol by other axis nodes. If an axis node

N fails the authentication protocol initiated by a specific axis node, it broadcasts a message to axis nodes in the DOMINO network informing them that axis node N might be compromised.

E.3 Obfuscation

Threat: A compromised node sends data (perhaps large amounts) that is supposed to be real in an attempt to obfuscate some other activity.
Remedies: There are two remedies for this threat. First, nodes attach SHA-1 digest with each block of data. The collision resistant property of SHA-1 will make it very hard for the adversary to tamper with the data sent by an axis node. The second remedy stems from the distributed nature of DOMINO. When results are forwarded between axis nodes, filters can be applied during the data fusion process such that no single node has the ability to skew results through simply increasing data volume. Filtering within a node set below an axis node can also be applied at the discretion of the axis node. The effect will be the same as the axis level filter. For obfuscation attacks not based on volume, the fusion process is designed to emphasize the *coordinated perspective* which significantly reduces or eliminates the effectiveness of this attack.
Threat: Attempts at stealthy and/or coordinated scanning.
Remedy: Perhaps the most important strength of DOMINO is the enhanced perspective afforded through coordination of multiple sites. This enhanced perspective can expose both stealthy and coordinated scans at much finer granularity than detection at a single site. However, if the adversary is willing to sufficiently slow their scanning or employs sufficiently many nodes in a coordinated fashion, they could still elude detection in DOMINO. The remedy is to include enough nodes in DOMINO to make the threshold on stealthy or coordinated scanning high enough to render this alternative infeasible.
Threat: An attempt to avoid active sink nodes.
Remedy: The most basic function of active sink nodes is to track scanning activity on *unused* IP addresses. In this sense, they will always be useful even if some adversaries can isolate their use to specific networks or IP addresses within networks. The combined use of an NIDS (on live IP addresses) and active sinks (unused IP addresses) will mean that all intrusion attempts have the possibility of being tracked. A simple way to confuse active sink identification is to employ probabilistic responses. Namely, instead of responding to all SYN packets in an IP block, only respond to some number of them.

We believe that as long as attackers spoof source addresses and active sink nodes monitor significant fraction of the unused IP space, traffic captured by the sinks will provide valuable insight into network intrusions.