NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
NWU-CS-04-40
July 12, 2004

Cyber Disease Monitoring with Distributed Hash Tables: A Global Peer-to-Peer Intrusion Detection System

Yan Chen **Aaron Beach** **Jason Skicewicz**

Abstract

Traffic anomalies and distributed attacks are commonplace in today's networks. Single point detection is often insufficient to determine the causes, patterns and prevalence of such events. Most existing distributed intrusion detection systems (DIDS) rely on centralized fusion, or distributed fusion with unscalable communication mechanisms. In this paper, we propose to build a distributed IDS based on the emerging decentralized location and routing infrastructure: distributed hash table (DHT). We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same sensor fusion center (SFC) while evenly distributing unrelated alarms to different SFCs. This is achieved through careful routing key design based on: 1) analysis of essential characteristics of three common types of intrusions: DoS attacks, port scanning and virus/worm infection; and 2) distribution and stability analysis of the popular port numbers and those of the popular source IP addresses in scans. We further propose load-aware node bootstrapping to distribute the alarms more evenly across the fusion centers. Evaluation based on one month of DShield firewall logs (600 million scan records) collected from over 2200 Worldwide providers show that the resulting system, termed Cyber Disease DHT (CDDHT), can effectively fuse related alarms while distributing unrelated ones evenly among the SFCs. Open questions on querying and attack-resilience of CDDHT are also discussed.

Keywords: security, distributed intrusion detection systems, distributed hash table

Cyber Disease Monitoring with Distributed Hash Tables: A Global Peer-to-Peer Intrusion Detection System

Yan Chen
ychen@cs.northwestern.edu

Aaron Beach
a-beach@northwestern.edu

Jason Skicewicz
jskitz@cs.northwestern.edu

Department of Computer Science, Northwestern University
Evanston, Illinois 60201
(12 pages in total)

Abstract—**Traffic anomalies and distributed attacks are commonplace in today's networks. Single point detection is often insufficient to determine the causes, patterns and prevalence of such events. Most existing distributed intrusion detection systems (DIDS) rely on centralized fusion, or distributed fusion with unscalable communication mechanisms. In this paper, we propose to build a distributed IDS based on the emerging decentralized location and routing infrastructure:** *distributed hash table (DHT)***. We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same sensor fusion center (SFC) while evenly distributing unrelated alarms to different SFCs. This is achieved through careful routing key design based on: 1) analysis of essential characteristics of three common types of intrusions: DoS attacks, port scanning and virus/worm infection; and 2) distribution and stability analysis of the popular port numbers and those of the popular source IP addresses in scans. We further propose** *load-aware node bootstrapping* **to distribute the alarms more evenly across the fusion centers. Evaluation based on one month of DShield firewall logs (600 million scan records) collected from over 2200 worldwide providers show that the resulting system, termed** *Cyber Disease DHT (CDDHT)***, can effectively fuse related alarms while distributing unrelated ones evenly among the SFCs. Open questions on querying and attack-resilience of CDDHT are also discussed.**

I. Introduction

Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network/service operators. It was estimated that malicious code (viruses, worms and Trojan horses) caused over $28 billion in economic losses in 2003, and will grow to over $75 billion in economic losses by 2007 [1].

The current state of the art in intrusion detection research is to use a combination of network-based intrusion detection systems (NIDS) and host-based intrusion detection systems (HIDS) to protect computer systems from compromise [2], [3]. Many of these systems suffer from high false positive and false negative rates due to their limited view of the global network. For instance, single point detection is often insufficient to determine the presence of a worm operating in the wild [4].

To this end, distributed IDS (DIDS) are purposed to leverage the diversity and strengths of existing third-party IDS technology to a distributed architecture in order to make global decisions about attacks observed in disjoint locations throughout the world [5], [6], [7], [8], [9], [10], [11], [12], [13]. Each IDS generates the attack symptom report based on its local detection, and sends to a global decision center, termed as *sensor fusion centers* (SFC). The SFC will correlate and analyze the prevalence, cause and patterns of the attack on a global scale. However, existing DIDS rely on centralized fusion techniques (Take a hierarchical DIDS for example, the central fusion point is its root) or distributed fusion techniques with un-scalable communication mechanisms (*e.g*, flooding to every SFC).

The exponential growth in the number of anomalies/attacks in the Internet demands a scalable distributed IDS infrastructure that is able to

- Effectively route alarms related to different intrusion events to different SFCs without consulting any central directory server or flooding mechanisms between peering points
- Integrate heterogeneous anomaly/intrusion detection systems in order to enhance detection diversity
- Support distributed queries over multiple fusion

centers to aggregate similar information
- Load balance across the multiple available fusion centers
- Achieve robustness and fault-tolerance in a network that is constantly evolving and failing
- Achieve resiliency in the face of attack

To address these challenges, we propose to build a distributed IDS fusion system based on the emerging distributed hash table (DHT) technologies. Systems built upon DHT technology [14], [15], [16], [17] provide decentralized routing and location infrastructure that may be useful to many applications, such as p2p file sharing [18], [19], information retrieval [20], content distribution networks [21], and even streaming media [22].

Weaver, Paxon, Staniford and Cunningham have suggested to build a Cyber Disease Control Center (CDCC) to defend against worms [4]. Here we name our DHT-based distributed IDS fusion system the Cyber Disease DHT (CDDHT), which embeds intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same fusion centers. From the routing of similar events to appropriate fusion centers, a global view is obtained to understand the prevalence, cause, and patterns of such attacks.

DHT provides some very nice properties for distributed systems, such as fault-tolerance, robustness [14], [15], [16], [17], and DoS attack resilience [23]. However, there are several challenges to properly design and implement the CDDHT system.
- How to design the routing key so that all related alarms are fused to the same SFC, while reducing other irrelevant alarms?
- How to achieve load balancing among the SFCs?
- How to support queries regarding intrusions?
- How to correlate intrusion alerts from heterogeneous IDSs, and make inferences?
- How to provide attack resilience when some set of IDS(s) and/or SFC(s) are compromised?

This paper provides some preliminary solutions and an evaluation to address the questions above. We will have more complete answers (and possibly more questions) by the time of workshop, and we hope this paper can stimulate interesting discussions regarding distributed IDS systems. In particular, this paper makes the following contributions.
- Based on their essential characteristics, we design the routing key (termed *disease key*) for three common types of intrusions: DoS attacks, port scanning and virus/worm propagation.
- With one-month of DShield firewall logs data consisting of over 600 million scan records from over 2200 providers around the world, we are among the first to study the popularity dynamics and distribution of top port numbers and top source IP addresses of scans. Such analysis is leveraged to design a proper disease key with a minimum length appropriate for the size of the DHT.
- We propose *load-aware node bootstrapping* to distribute the alarms more evenly across the SFCs.

We have built the CDDHT system on top of the Chord DHT [15], and evaluate it by simulation with daily DShield firewall scan logs involving over 1400 providers of over 25 million scan logs. The results show that we can effectively fuse the related alarms while distributing unrelated reports evenly among the SFCs. Open questions on querying and attack-resilience are also discussed.

The rest of the paper is organized as follows. In Section 2 we discuss work related to this project. Section 3 provides an overview of the architecture of CDDHT and the role of a peer node within our system. Section 4 discusses various design aspects of CDDHT system, including disease key, load balancing, querying, robustness and attack resilience. We evaluate the CDDHT system in Section 5, and finally conclude the paper in Section 6.

II. Related Work

A. Distributed Intrusion Detection

Early work that appeared on distributed intrusion detection systems occurred in the early 1990s [5]. This system combined distributed monitoring and data reduction with centralized data analysis. NSTAT [7], part of the STAT (State Transition Analysis Technique) tool collection, uses state transition analysis over distributed sensors that forward their data to a central entity. This central entity merges the audit trails from each of the sensors into a chronological snap shot of each of the event transitions that occurred in the network. Each of the above systems suffer from the vulnerability of single central node making global decisions, and also does not scale with the number of events.

Because of the single point of failure and scalability issues specific to the previous distributed intrusion detection systems, development of hierarchical variants were created. GrIDS [24] is an IDS built to detect distributed attacks against large networks. In this system, instead of simply forwarding all audit data collected from end hosts and network to a collector, the idea here is the construction of activ-

ity graphs that represent hosts and network activity between them. The graph engines can be deployed in hierarchical fashion, where higher level engines receive sub-graphs generated from lower level modules. EMERALD [8] is an intrusion detection system from SRI that combines anomaly and misuse detection mechanisms, focusing on providing a system for large-scale enterprise networks. The networks themselves can be divided into independent domains that exhibit different trust relationships and security policies. AAfID [9], the Architecture for Intrusion Detection using Autonomous Agents employs a hierarchical design to combat the disadvantages of a central, monolithic architecture. The design goals of this architecture are to improve system configurability and the resistance against denial of service attacks.

Other notable work, include publications by Kruegel and Toth which cover a wide spectrum of topics in distributed IDS [10]. Problems with much of this work, is that their architecture is based on a peer-to-peer system that passes events towards a root node that makes the final decision. If one of the nodes in the chain is compromised, it can feed events to the root node that obfuscate the detection process.

Related work that is closest in spirit to our work here include Indra [11], DOMINO [12] and DNAD [13]. Indra is a distributed scheme of intrusion detection and prevention based on the sharing of information between trusted peers in a network. Its prototype implementation is built upon the Scribe [25] project, which overlays a topic-based publish-subscribe multicast mechanism on top of the pastry peer-to-peer network [16]. DOMINO is an architecture for a distributed intrusion detection system that fosters collaboration among heterogeneous nodes organized within an overlay network. This work builds upon an earlier study [26] which finds that blacklisting specific source IP addresses can provide much benefit, as many attacks come from some top N sources. In the DOMINO overlay system, every node maintains a local and global view of intrusion and attack activity. Local views consider activity in its own network, and periodic summaries are shared between peers which are then used to create a view of global activity. In the DNAD project, they push the idea of "collaborative security" which they believe provides greater clarity about attacker intent, precise models of adversarial behavior and a better view of global network attack activity. Their proposal is a decentralized system that can efficiently distribute alerts to collaborating peers. They use a tool for extracting relevant information from alert streams and another for scheduling correlation relationships between peer nodes.

Our system is similar in spirit to the above, but we concentrate here on the design of routing events to appropriate sensor fusion centers. We use sensor fusion techniques for gathering a global view into distributed attacks over the network. DNAD and Indra deal with trust agreements, whereas our system hopes to deal with trust statistically at the SFCs. DOMINO uses a small set of axis nodes for obtaining the global view, whereas we envision much greater scale whereas users of the internet can deploy a firewall and send events to our SFCs.

Our proposal in using third-party IDS systems whose events map to a single output data format, and communicate using peer-to-peer mechanisms alleviate many of the problems in both central and hierarchical designs. In addition, even hierarchical approaches, while alleviating the single point of failure and scalability issues, are fixed in the number of levels and may present configuration issues. In our work here, we provide a general framework for sensor fusion. Because of the properties exhibited by the DHT mechanisms that we employ for routing, the SFCs are able to achieve self-organization, self-healing, fault-tolerance, scalability and robustness in the face of attack. Our system uses this distributed hash table (DHT) technology in order to route messages to a sensor fusion center. Each orthogonal attack maps to a different SFC, and upon failure, a near neighbor takes over the responsibility for fusing the event data.

B. DHT and Peer-to-Peer Systems

Distributed hash table (DHT) technology is used in projects such as CAN [14], Chord [15], Pastry [16] and Tapestry [17]. Each provide a scalable and robust data location and request routing service for large-scale distributed applications. A node participating in an overlay is assigned a random identifier from a numeric space. All DHT-based systems export an interface for insertion (*put*) and retrieval (*get*). The interface for insertion, `put(key,object)`, causes the DHT to route the given `object` to the node with a node identifier closest to the `key`. The interface for retrieval, `object=get(key)`, causes the DHT to obtain the `object` from the node with a node identifier closest to the `key`.

In our work we use distributed hash tables to decentralize the locations of sensor fusion centers, pro-

vide fault-tolerance, scale the system in the number of deployed intrusion detection systems, and scale in the routing of messages throughout the peer-to-peer system. Our system uses an *attack symptom report* as the `object` of the interface and the *disease key* as the `key`. The generation of the attack symptom report and the design of the disease key will be discussed in detail in Sections 3 and 4.

By decentralizing our sensor fusion centers, we ensure that the number of messages that each center receives is specific to a certain attack symptom observed in the network. And there is no centralized directory service. Fault-tolerance is extremely important as the sensor fusion centers themselves may be subject to compromise. As an SFC is compromised, a new SFC will take over the responsibility of making global decisions through self-organization algorithms which exist as part of a DHT based peer-to-peer system. Scalability in the deployment of a global IDS and in the number of messages is also important in that global information requires a large number of sensors and wide diversity of sensors in the types of attacks that each can detect. These properties of DHT based peer-to-peer systems are equally important to a globally distributed intrusion detection system.

III. System Architecture

The architecture of the CDDHT system is shown in Figure 1. There are multiple heterogeneous IDS sensors connected through DHT. A subset of them with more resources (*e.g.*, more CPU and bandwidth) and better security measures are selected as SFCs. SFCs are recognized by their DHT node IDs, *e.g.*, the first few bits of their node IDs are predetermined. The amount of IDSs and especially the router-based IDSs, in CDDHT determines the overall coverage of the Internet used for gleaming the presence of widespread attacks. When an attack is launched, each IDS peer attempts to locally detect the attack. Upon detection, it generates a symptom report that will be forwarded to the appropriate SFC, determined by the symptoms and inferred attack type. The node IDs of SFCs correspond to the disease key which can best characterize the intrusion events so that all related alarms of an intrusion will be routed to the same SFC. For instance, in Figure 1, there is a distributed attack detected by two IDSs, and both send alarms to the same SFC. For an attack with *a prior* disease key, we can query its prevalence by routing to the corresponding SFC.

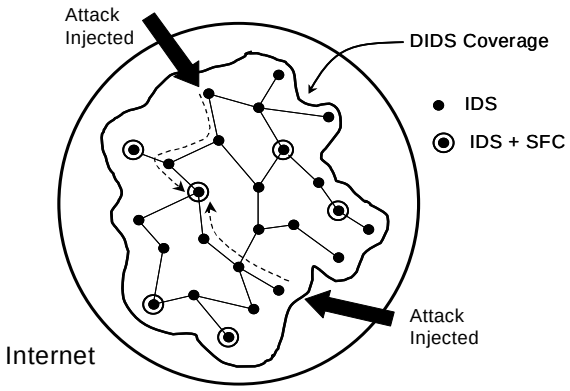

Fig. 1. Architecture of the CDDHT system.

A. Symptom Report

There are various types of IDSs deployed in the Internet. According to the information source, there are network-based IDSs, host-based IDSs, and application-based IDSs. IDSs can be classified as *anomaly detection* which use statistical methods for inferring anomalous behavior and *signature detection* which use pattern matching to find existing profiled attacks. Good classification of IDS systems can be found in [2], [3], [27].

Each type of detector has its own strengths and weaknesses. Through the deployment of heterogeneous intrusion detection systems throughout the Internet, we obtain diversification in the number of systems and leverage the strengths of each. Each IDS system standing alone may generate high false negative and false positive rates. By aggregating similar events generated from each IDS to distinct sensor fusion centers, we hope to minimize these rates.

In order to use the information generated by these heterogeneous IDSs, a common output format is desired. Existing IDSs have proprietary and often incompatible output event formats. However, there are many standardization efforts in the works, but it is unclear whether these have reached the involvement necessary for widespread, diverse deployment of third-party systems. Two typical such efforts include Intrusion Detection Message Exchange Format (IDMEF) proposed by the Intrusion Detection Working Group (IDWG) [28] and the Common Intrusion Detection Framework (CIDF) proposed by DARPA [29]. We leverage them to fuse the correlated events at the fusion centers.

IV. Cyber Disease DHT (CDDHT)

As events are generated locally at peer IDS nodes, the symptoms of an attack must be reported and

routed to the sensor fusion center which can then make inferences about the global prevalence, causes and patterns of such attack. In this section, we construct the Cyber Disease DHT for such routing, and discuss the routing key design, load balancing, querying and attack resilience of CDDHT.

A. Disease Key for Intrusion Event Aggregation

As introduced in Section 2, there are two basic operations for DHT: *put* and *get*. We will use the `put(key,object)` operation for sending symptoms to SFCs. Thus `object` is the attack symptom report, and `key` is the the routing ID which is generated from the report, and can deterministically map to a node on DHT. We term it as *disease key*.

The challenge is: for a single intrusion, given the vast diverse symptoms perceived from many heterogeneous IDSs around the world, how to design the key to effectively fuse these events to a SFC? For instance, while a worm is propagating, a router-based IDS may see much larger ranges of source/destination IPs/ports along with a higher scan rate than that seen at a host-based IDS. For a distributed DoS attack, the network IDS (NIDS) for a zombie subnet; the NIDS for a victim subnet; and the NIDS for a third-party subnet have completely different views of attack and response traffic as either ingress, egress, or both.

Furthermore, the design should achieve as even a distribution across the SFCs as possible for unrelated symptoms while still ensuring aggregation of related symptoms along with maintaining fault tolerance. All of these properties must be accomplished through key generation by each peer IDS in a decentralized and deterministic manner.

It is possible that multiple routing IDs ought to be derived for each type symptom report, based on different criteria, so as to allow detection of a greater variety of global events through a more broad distribution of information. The DHT will effectively route the data to whichever nodes the routing ID hashes to. Therefore it is evident that the use of an intelligent hashing process is absolutely necessary to the proper functioning of our routing protocol. The data that is used to generate the routing ID should be as minimal and concise as possible in order to ensure confidence that related event symptoms will be routed to the same SFCs. The ideology behind choosing the fields for the routing ID was: to use only the information that uniquely identified related events. Extraneous identifiers that may have clarified routing in some cases but confused in others were thrown out. A minimal besteffort solution was selected. The following design is based on the following sources: 1) survey of various attacks: worms [30], [31], [32], denial of service attacks [33], [34], [35], [36] and port scans [37], [38]; and 2) study of the Internet intrusion characterization based on one-month DShield [39] data of over 600 million scan records.

The format of the disease key is shown in Table 2. The first element of the routing ID is a two bit identifier differentiating between the three major types of activity we wish to gain a global understanding of. This element of identification may be expanded for actual implementation in order to allow extensibility in the future. The three major types of activity we focus on are viruses/worms, scans, and DoS attacks, as discussed next.

Ideally, the disease key for DoS should properly distinguish and correlate millions of DoS attack events, but DoS attacks are very hard to characterize [33]. There may be one or many attack agents sending DoS attack traffic, and these agents often have IP address spoofed, or they may be well-known servers sending legitimate response (*e.g.*, distributed reflection attacks [40]). Other than application-specific DoS attacks (*e.g.*, DNS request attack, a bogus signature attack on an authentication server, etc.), DoS attacks do not have fixed port number. The TCP is the most common protocol in DoS, due to the SYN flooding attacks. But the more recent attacks can be a mixture of TCP, UCP and ICMP traffic, like TFN, TFN2K [41], Stacheldaraht [42] and Shaft [43]. The attack rate can be constant or variable, and even the zombie agent set for the same attack may receive different commands from the attacker, and thus have different attack behavior throughout the attack [33].

Thus the only invariable for a DoS attack is the victim, which is used in our design of the disease key. For application or host based attacks, the victim IP address is used as the key. For network attacks, which consume the bandwidth of a target network subnet, the key is the subnet (longest prefix) with all remaining bits set to 0. Note that the victim has to be recognized by each IDSs, and the ”victim” IP address can be source or destination depending on the situation. Take a TCP SYN flooding attack for example, for the IDSs close to the attack agent or the victim machine, the victim is the destination IP address while for the thirdparty IDSs which receives “backscatter” traffic [34], the victim is the source IP address. Given any DoS attack detection there must

Intrusion	ID	Characterization Field(s)		
DoS Attack	0	Victim IP (subnet)		
Scans	1	0 (for vertical and block scan)	Source IP address	Destination IP (for vertical scan)
				0 (for block scan)
		1 (for horizontal and coordinated scan)	Scan port number	Source IP (for horizontal scan)
				0 (for coordinated scan)
Viruses/Worms	2	0 (for known virus/worm)	Worm ID	
		1 (for unknown virus/worm)	Destination port number	

Fig. 2. Disease Key of CDDHT

be one or a few most prominent addresses (victims) that are being overwhelmed.

Scan is probably the most common and versatile type of intrusion. Based on source/dest IP and port number involved in the scans, there are four well known types of scans: *horizontal scan*, *vertical scan*, *block scan*, and *coordinated scan* [26], [44]. Horizontal scan is the most common type scan, which scans the port of interest on all IP addresses in some range of interest. We denote it as an identifier bit 1 and the scan port number. The port number is unique because it reflects the vulnerability the virus/worm try to explore. Unlike DoS attacks, the attacker does need to use a real source IP address, since she needs to see the packet that servers generate in response to the scan in order to know what ports are actually open [44]. A recent Internet intrusion study on its global characteristics and prevalence also found that the popularity of the attacker source IP address follows heavy-tail distribution which further confirms that real IP address is used in scan, and it can be help characterize different scan events [26]. Thus we use the source IP address as the last field of the scan disease key.

A vertical scan is a scan of some of all ports on a single host, with the rationale that the attacker is interested in this particular host, and wishes to characterize its active services to find which exploit to attempt, or to find a suitable exploit via her network of contacts and resources [44]. We represent this scan with the identifier bit 0, and the source IP and destination IP addresses in the disease key. We do not include the port set because such set are not stable and are hard to represent accurately with small space in disease key.

A block scan is a combination of horizontal and vertical scans over numerous services on numerous hosts [44]. This scan can be regarded as vertical scan without fix destination. We use the same disease key as the vertical scan except the destination IP address as all 0 because of its variance. The fourth type of scan, coordinated scan, can be viewed as horizontal scans from multiple sources [26]. We use the same disease key as the horizontal scan except that the source IP address is set to zero because of its intrinsic variance in a coordinated attack. More complicated scans than these four are possible in principal, but we have not seen much in practice, and for now we leave them as subsets of these major classifications.

Most worms/viruses are detected through the signatures of existing viruses or worms. For each type of worms/viruses, we designate a global unique ID. The disease key for worms/viruses consists of an identifier bit 0 followed by the ID. Normally, unknown viruses or worms are detected through their scanning as discussed before. For unknown virus or worm detected through anomaly detection, we denote it with an identifier bit 1 and the intrusion (destination) port number associated with the attack.

B. Dimension Mismatch between Disease Key and the DHT

We have carefully chosen the characteristics to compose the disease key. However, it is still unclear how these characteristics can be combined as a good routing ID on DHT. Can we simply concatenate and/or hash these fields as the disease key to get proper aggregation of intrusion events with even distribution? The answer is no, with the analysis below and also confirmed by our simulation results in Section 5-C.

The keys are of variable length for the design above: viruses/worms may only need 19 bits, while the key for vertical scans can have up to 67 bits. For most DHT systems [15], [16], [17], given a n-node DHT, the key length is $\log n$. Then even a 10^6 node DHT can only have 20-bit key. We can increase the key length by a constant to accommodate all keys. But the bigger the constant, the more virtual nodes one real DHT node has to cover. To support the aggregate queries discussed in Section 4-D, we cannot hash the key randomly to distribute the load. Then a large key space can easily lead to unbalanced load distribution.

Some load balancing schemes, such as the load-aware node bootstraping in Section 4-C, are independent to the disease key design. But an unnecessarily large key space makes it hard to balance, and wastes extra resources for neighbor table maintenance. Since port scanning is probably the most common type of intrusion, and has the longest key, we will study how to design a key of the minimum length, but can still differentiate the majority of scans. The distributions of the scan port number and source IP address are found to have uneven distributions, *i.e.*, a small number of entities are responsible for a large number of scans [26]. This suggests that as long as we can represent a small number of popular ports with different keys, the hot spots will be evenly distributed to different nodes. Then the number of bits needed is much less than 16 for ports or 32 for IP addresses. However, how long do the popular ports remain popular? If they keep changing, the hot spots may still get collided to the same SFC.

To answer these questions, we study the distribution and stability of the popular scan ports, and those of top source IP addresses. We consider two metrics for the stability: i) the stability of its popularity ranking, *e.g.*, whether top 10 remains to be top 10; and ii) the stability of the scan record coverage from previous popular entities, *e.g.*, how many scan records of Jan. 10 are covered by the top 10 ports of Jan. 1st. These two metrics do not always agree with each other. For instance, we found that while the ranking stability of popular ports fluctuates a lot, but the coverage stability is very stable.

Our analysis is based on the real one-month (January 2004) intrusion logs from DShield [39] consisting of over 600 million scan records from over 2200 providers around the world. We analyze the stability in various time scales: a day, a week, two weeks, three weeks, and four weeks.

B.1 Distribution and stability of the top scan port numbers

As shown in Figure 3, for each day in the month, 90% or more of total scans are covered by 64 most popular ports in that day [1]. Figure 4 and Figure 5, respectively, show the ranking and coverage stability for the most popular ports. It should be noted that while ranking stability fluctuates, scan coverage continues to rise for all days surveyed. Therefore this shows that given a sufficient training time period (one day in our example) the most popular ports will cover a large percentage of scans in the future. For instance, 64 popular ports remain to cover 85% of the scan records for up to three weeks. So with good hashing function, only 6 bits in the disease key are sufficient to represent and distinguish most popular port numbers.

[1]We only show a subset results of the 31 days in the month.

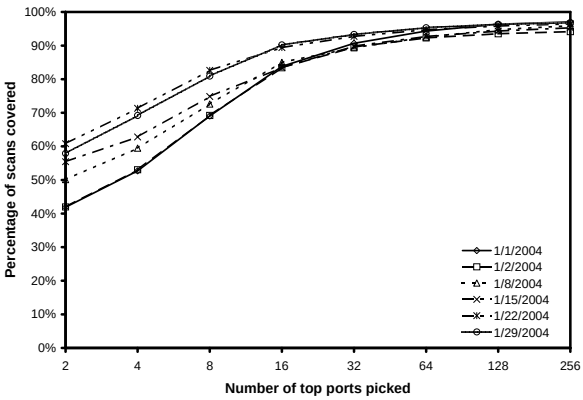

Fig. 3. Cumulative percentage of the scan records covered by top ports

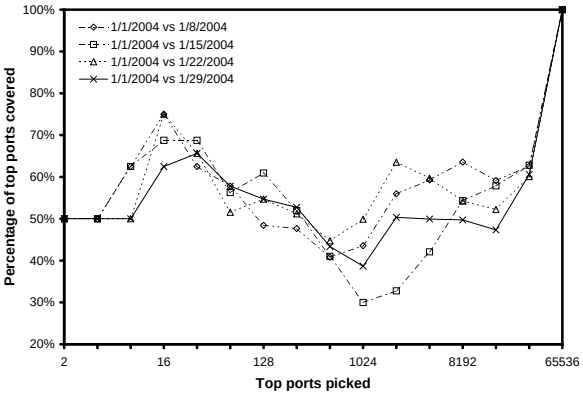

Fig. 4. Ranking stability of the most popular ports chosen at 1/1/2004

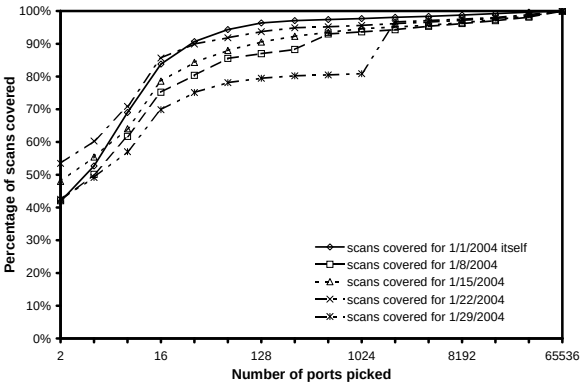

Fig. 5. Scan coverage Stability of the most popular ports chosen at 1/1/2004

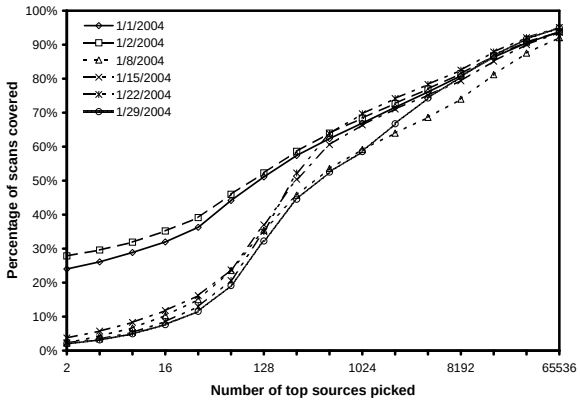

Fig. 6. Scan distribution across most popular sources each day

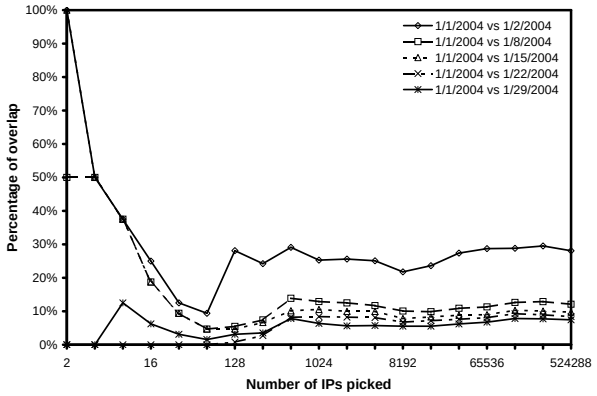

Fig. 7. Ranking stability of the most popular source IP addresses of 1/1/2004

B.2 Distribution and stability of the top scan source IP addresses

Figure 6 shows that, like ports, the most popular scan sources account for a large percentage of the total scans. However, as seen in Figure 8 and Figure 7, the stability of the first day's most popular sources drops over time to less than 10% coverage by either

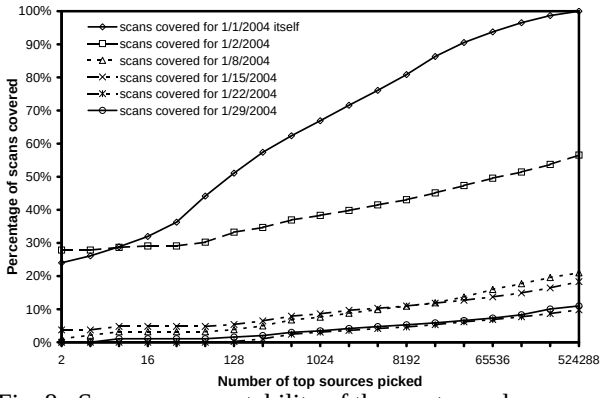

Fig. 8. Scan coverage stability of the most popular source IP addresses of 1/1/2004

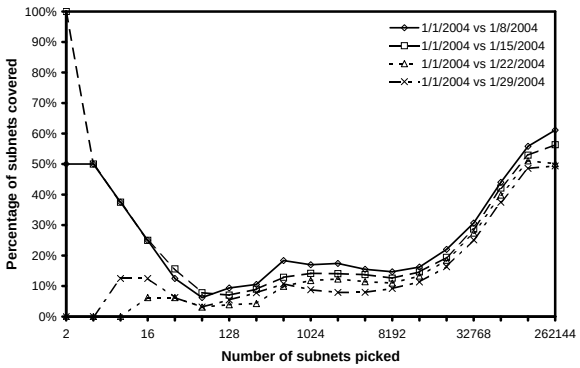

Fig. 9. Ranking stability of the most popular subnets of 1/1/2004

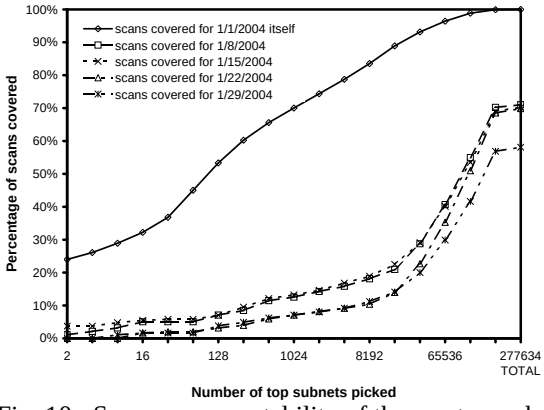

Fig. 10. Scan coverage stability of the most popular subnets of 1/1/2004

metric. So, unlike port numbers, the popular source IP addresses of scan keep changing. That is, there is no steady blacklist.

We then study the stability of the class C (/24) subnets of source IP. The results as shown in Figures 9 and 10 have similar trends as those of source IPs. That is, there is no particular subnets are continuously being used for attacks. Thus in the disease key of horizontal and coordinated scans, we use six bits for port, and use all remaining bits to represent source IP addresses. For vertical and block scans, the source and destination IP addresses have the same length in the disease key.

C. Load Balancing

Some of the field values in the disease key are more frequently used by other fields values. For example, Figure 11 shows how ports with lower numbers or ranges that include common scan ports (128-256) are much more popular scan targets than other ports. Even among the ports with number less than 1024, the popularity distribution is not even: ports 21 (FTP), 53 (DNS), 80 (HTTP), and 137 (NETBIOS)

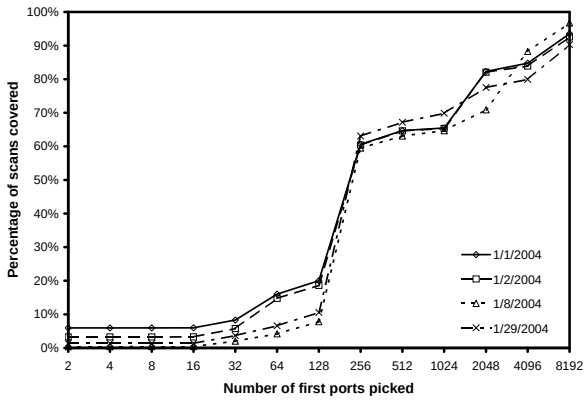

Fig. 11. Scan coverage distribution across the first 8192 ports

are the hot spots [45].

We have two solutions for load balancing: 1) hash the port number before adding it to the key; and 2) *load-aware node bootstrapping*. Hashing can be further leveraged on the popular port stability we studied before, and design special hashing functions to evenly distribute the most popular ports from a certain "training" time period.

Load-aware node bootstrapping happens when we add an IDS node or any other node as a SFC. Each time when an IDS send alarms to a SFC, the SFC will reply as confirmation, and piggyback its recent load (*e.g*, number of reports received in the past 24 hours). Then an IDS becomes a SFC, it will contact the SFC with the heaviest load which will split in the middle along the busiest and divisible dimension, and hands over the half of its zone to the new SFC. When a new node joins as a SFC, it will consult its nearby IDS for the heavy load SFC, then do the same.

Next, we discuss a few open questions on CDDHT.

D. Querying on CDDHT

On getting symptom reports from various places, SFC will correlate and merge alerts with event fusion algorithms like [46], [47]. The attack summary is sent to the interested users (*e.g.*, the IDSs which reported related symptoms), or is used to answer queries.

CDDHT can support two types of queries: with a given disease key or *aggregate queries*. The former is easy: simply calling `get(disease key)`. The latter can answer queries like "show me all the horizontal and coordinated scans with port number x". This is particularly useful to detect a complicated worm propagation where some nodes perceive it as a horizontal scan while other nodes perceive it as a coordinated scan. The query is issued to the SFC which is responsible for the coordinated scan. That SFC will serve as collector, and query all other nodes in the "zone" which have the same node ID as the collector except the last a few bits for source IP address.

We also plan to leverage "range queries" over DHT [48] for more complicated query requests.

E. Robustness and Attack-Resilience

When any SFC leaves the CDDHT or just becomes unavailable, other nearby CDDHT node(s) [2] will take over its responsibility [14], [15], [16], [17]. We can also replicate each SFC node to make it more robust.

Some of the IDS sensors may get compromised and send out bogus attack information to SFC. We use the voting mechanisms similar to DOMINO [12] to ignore those faked alarms. However, if the SFC is compromised, it will not be able to report the corresponding type of intrusions, but other SFCs can still work without being affected. Furthermore, since we hash some of the key fields like the port number, given an intrusion, the attacker does not know its disease key, and thus does not know the corresponding SFC. We are investigating other techniques to further improves the attack-resilience.

V. Evaluation

In this section, we present the simulation methodology and results for our preliminary evaluation on CDDHT system.

A. Methodology

As shown in Section 4-B, our simulation is based on daily DShield firewall logs data. They are only denoted as scans, so we will focus on evaluating the effectiveness of disease key for scans, which arguably represents the largest number of intrusions in the current Internet. It is our future work to collect data on DoS attacks and virus/worm infection for a more comprehensive evaluation of the whole system.

For the logs of each day, first we generate the four types of events for the providers, with definition mostly adopted from [26] and [44] as below.

- **Vertical scan** We define vertical scans as one or more scans from one source to one destination (victim) across five or more ports.

[2]They are logically close. So these nodes have similar node IDs, and are also SFCs.

• **Horizontal scan** We define horizontal scans as one or more scans from one source to five or more destinations (victims).
• **Block scan** While horizontal and vertical scans vary in one dimension (destination or port), a block scan varies in both dimensions (destination and port). Therefore we define a block scan as an combination of vertical and horizontal scans. This combination is seen as scans from one source covering five or more of the same ports on each of five or more destinations (victims).
• **Coordinated scan** We define coordinated scans as 5 or more sources conducting parallel horizontal scans to 5 or more destinations within the same subnet. We do not account for coordinated vertical scans, reporting them as multiple unique vertical scans because it rarely happen in practice.

Note each scan event only count scan logs from the same provider. For each scan log record, we classify it to *only* one of the four types of scans. For example, if one horizontal scan is detected as part of a coordinated scan, it will only be reported as part of the coordinated scan, but not as a horizontal scan.

We then used these classified scan events to generate the disease keys based on the design in Section 4. Each of the disease key is simulated to route on the Chord [15] DHT. Before that, Chord is initialized with the n nodes where n is the number of unique providers (IDS nodes) in the DShield data. For simplicity, we assume any DHT node (*i.e.*, IDS node) can be a SFC.

B. Metrics

The metrics include the effectiveness of fusion, and the variation of load among the SFCs. The former is the percentage of related alarms (defined by the disease key) fused in the corresponding SFC. For each SFC, the load is denoted by the number of symptom report received. The load variation is measured in terms of the *coefficient of variation* (CV) and the *maximum vs. mean ratio* (MMR). The CV of a distribution x, defined as below, is a standard metric for measuring inequality of x, while the MMR checks if there is any single node whose load is significantly higher than the average load.

$$CV(x) = \frac{standard\ deviation(x)}{mean(x)} \qquad (1)$$

C. Simulation and Results

We use daily DShield data for evaluation, and they all have similar results. So we will just show the results for that of January 2nd, 2004. It contains over 25 million scan logs from nearly 1417 providers. These scan logs are classified into the events as in following table.

scan type	Vertical	Horizontal	Block	Coordinated
# of scans	3364	8486	22	25711

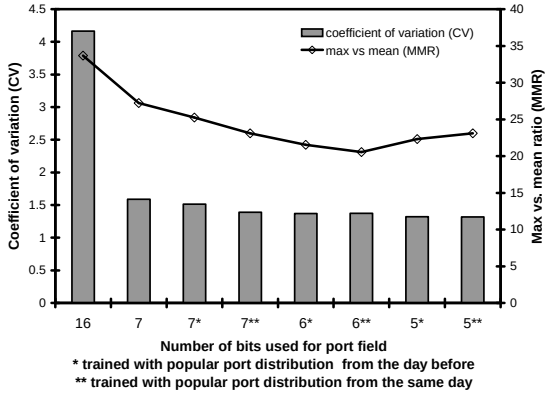

Fig. 12. Statistical Distribution of reports across the DHT nodes

The large number of coordinated scans is probably caused by the large amount of worms spreading nowadays. With many worms propagating between subnets, the same port is often scanned by many different sources.

For Chord, we use 32-bit node ID and 32-bit routing key by default. The stability study in Section 4-B suggests using 6 bits to represent port, we further compare it with 16 bits full length, and with little variation for 5 and 7 bits as below. The source IP addresses are quite random and do not have any popularity stability, so we just use the IP address (starting from the most significant bit) to fill the remaining bits in the key.
• hashing but maintaining 16 bit field length
• hashing to 7 bits
• hashing to 5,6 or 7 bits while using a hash function trained with the most popular ports from the day before (Jan. 1 2004)
• hashing to 5,6 or 7 bits and using hash function trained with the most popular ports of the day being tested (Jan. 2 2004)

By training a hash function, we manually set the the top 2^x ports with different x-bit representations where x is the number of bits to represent the scan port number in the key. This is to ensure these top ports are evenly distributed with the x bits.

In terms of fusion, for all the events, all the related alarms are fused to its corresponding SFC because Chord has deterministic routing, and thus given the

same disease key, it always routes to the same node.

The load distribution results are shown in Figure 12. To reduce the number of bits for port number in the key can significantly improve the load balancing: it reduces the coefficient of variation by more than 60% and reduce the maximum vs. mean ratio by about 40%. Six-bit port number design gives the best load distribution, but 5 and 7 bits also have very similar results. There are still certain hot spots because some scan events are much more popular than the rest, as indicated in the MMR ratio. Load-aware node bootstraping can potentially solves that problem by having more SFCs for the particular hot event. We will evaluate its effectiveness as part of our future work.

With training hashing functions, Figure 12 shows that it helps improve the load distribution by about 15%. Since the popular ports remain very stable, there is little difference between training the hashing function with the history or with the current dataset (like the oracle case). We also tried training the hashing function with older datasets (up to a month old), which gave similar results.

VI. Conclusion

Most existing distributed intrusion detection systems (DIDS) rely on centralized fusion, or distributed fusion with unscalable communication mechanisms. In this paper, we propose to build a distributed IDS based on the emerging decentralized location and routing infrastructure: *distributed hash table (DHT)*. We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion (thus with similar symptoms) will be routed to the same sensor fusion center (SFC) while evenly distributing unrelated alarms to different SFCs. This is achieved through careful routing key design discussed in the paper. Evaluation based on massive worldwide firewall logs data show that the CDDHT system can effectively fuse related alarms while distributing unrelated ones evenly among the SFCs. Open questions on querying and attack-resilience of CDDHT are also discussed.

References

[1] E. Mars and J. D. Jansky, "Email defense industry statistics," `http://www.mxlogic.com/PDFs/IndustryStats.2.28.04.pdf`.

[2] Stefan Axelsson, "Intrusion detection systems: A survey and taxonomy," Tech. Rep. 99-15, Department of Computer Engineering, Chalmers University of Technology, Sweden, 2000.

[3] D. Alessandri et al., "Malicious and accidental fault tolerance for internet applications (maftia): Towards a taxonomy of intrusion detection systems and attacks," Tech. Rep. Research Report RZ 3366, IBM Research, 2001.

[4] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham, "Large scale malicious code: A research agenda," Tech. Rep., DARPA Sponsored Report, 2003.

[5] Steven R. Snapp et al., "Dids (distributed intrusion detection system) - motivation, architecture and an early prototype," in *Proceedings 14th National Computer Security Conference*, 1991.

[6] Tim Bass, "Intrusion detection systems and multisensor data fusion," *Communications of the ACM*, vol. 43, no. 4, 2000.

[7] R. A. Kemmerer, "A model-based real-time network intrusion detection system," Tech. Rep., University of California Santa Barbara, Technical Report TRCS97-18, 1997.

[8] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling responses to anomalous live disturbances," in *In 20th NIS Security Conference*, 1997.

[9] J. S. Balasubramanyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *In 14th IEEE Computer Security Applications Conference*, 1998.

[10] Christopher Kruegel, Thomas Toth, and Clemens Kerer, "Decentralized event correlation for intrusion detection," in *In International Conference on Information Security and Cryptology (ICISC)*, 2001.

[11] Ramaprabhu Janakiraman, Marcel Waldvogel, and Qi Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *In Proceedings of IEEE WETICE 2003 Workshop on Enterprise Security*, 2003.

[12] Vinod Yegneswaran, Paul Barford, and Somesh Jha, "Global intrusion detecction in the domino overlay system," in *The 11th Annual Network and Distributed System Security Symposium (NDSS)*, 2004.

[13] Michael E. Locasto, Janak J. Parekh, Sal Stolfo, Angelos D. Keromytis, Tal Malkin, and Vishal Misra, "Collaborative distributed intrusion detection," Tech. Rep. CUCS-012-04, Columbia University Computer Science Department, 2004.

[14] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A scalable content-addressable network," in *In Proceedings of ACM SIGCOMM 2001*, 2001.

[15] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnany, "Chord: A scalable peer-to-peer lookup service for internet applications," in *In Proceedings of SIGCOMM 2001*, 2001.

[16] Anthony Rowstron and Peter Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *In Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.

[17] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, 2004.

[18] Bram Cohen, "Incentives build robustness in bittorrent," `http://bitconjurer.org/BitTorrent/bittorrentecon.pdf`, May 2003.

[19] "Shareaza p2p, bringing p2p together," http://www.shareaza.com/.

[20] Chunqiang Tang, Zhichen Xu, and Sandhya Dwarkadas, "Peer-to-peer information retrieval using self-organizing semantic overlay networks," in *In Proceedings of SIGCOMM'03*, Karlsruhe, Germany, August 2003, pp. 175–186.

[21] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in a cooperative environment," in *In Proceedings of the IPTPS'03*, 2003.

[22] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker, "Application-level multicast using content-addressable networks," in *In Proceedings of NGC 2001*, 2001.

[23] Y. Chen, A. Bargteil, D. Bindel, R. Katz, and J. Kubiatowicz, "Quantifying network denial of service: A location service case study," in *Proc. of the Third International Conference on Information and Communications Security (ICICS)*, 2001.

[24] S. Staniford, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids - a graph based intrusion detection system for large networks," in *In 20th National Information Systems Security Conference*, 1996, vol. 1.

[25] Anthony Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel, "Scribe: The design of a large-scale event notification infrastructure," *In Networked Communication*, 2001.

[26] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich, "Internet intrusions: Global characteristics and prevalence," in *In Proceedings of ACM SIGMETRICS*, 2003.

[27] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, "State of the practice of intrusion detection technologies," Tech. Rep. CMU/SEI-99-TR-028, Carnegie Mellon University, 1999.

[28] IDWG, "Intrusion detection message exchange format," http://www.ietf.org/html.charters/idwg-charter.html, 2003.

[29] DARPA, "Common intrusion detection framework," http://gost.isi.edu/cidf, 1999.

[30] Stuart Staniford, Vern Paxson, and Nicholas Weaver, "How to own the internet in your spare time," in *In the Proceedings of the 11th USENIX Security Symposium (Security 2002)*, 2002.

[31] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham, "A taxonomy of computer worms," in *In Proceedings of the First Workshop on Rapid Malcode (WORM)*, 2003.

[32] David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver, "The spread of the sapphire/slammer worm," http://www.caida.org, 2003.

[33] Jelena Mirkovic and Peter Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *In ACM Computer and Communication Review (CCR)*, 2004.

[34] David Moore, Geoffrey M. Voelker, and Stefan Savage, "Inferring internet denial-of-service activity," in *In Proceedings of the USENIX Security Symposium*, 2001.

[35] Jaeyeon Jung, Balachander Krishnamurthy, and Michael Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites," in *In Proceedings of the WWW*, 2002.

[36] Aleksander Kuzmanovich and Edward W. Knightly, "Low-rate tcp-targeted denial of service attacks (the shrew vs. the mice and elephants)," in *In Proceedings of the ACM SIGCOMM*, 2003.

[37] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *In Proceedings of the IEEE Symposium on Security and Privacy*, 2004.

[38] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage, "Internet quarantine: Requirements for containing self-propagating code," in *In Proceedings of the IEEE Infocom*, 2003.

[39] DShield.org: Distributed Intrusion Detection System, ," `http://www.dshield.org/`.

[40] S. Gibson, "Drdos: Distributed reflection denial of service," 2002, `http://grc.com/dos/drdos.htm`.

[41] J. Barlow and W. Thrower, "Tfn2k - an analysis," 2000, `http://packetstormsecurity.com/distributed/TFN2k_Analysis-1.3.txt`.

[42] D. Dittrich, "The "stacheldraht" distributed denial of service attack tool," 1999, `http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt`.

[43] S. Dietrich, N. Long, and D. Dittrich, "An analysis of the "shaft" distributed denial of service tool," 2000, `http://home.adelphi.edu/~spock/shaft_analysis.txt`.

[44] S. Staniford, J. A. Hoagland, and J. M. McAlerney, "Practical automated detection of stealthy portscans," *Journal of Computer Security*, vol. 10, no. 1-2, 2002.

[45] DShield.org: Distributed Intrusion Detection System, "Top target ports," 2004, `http://www.dshield.org/port_of_the_day.php`.

[46] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *Proc. of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2001.

[47] F. Cuppens and A. Miege, "Alert correlation in a cooperative intrusion detection framework," in *Proc. of the IEEE Symposium on Security and Privacy*, 2002.

[48] Scott Shenker Sylvia Ratnasamy, Joseph M. Hellerstein, "Range queries over dhts," Tech. Rep. Intel Research Technical Report, IRB-TR-03-011, 2003.