Towards Scalable and Robust Distributed Intrusion Alert Fusion with Good Load Balancing

Zhichun Li, Yan Chen, Aaron Beach
Department of Electrical Engineering and Computer Science, Northwestern University
2145 Sheridan Road, Evanston, IL 60208, USA
{zli,ychen,a-beach}@northwestern.edu

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 DIDS 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 four common types of intrusions: DoS attacks, port scanning ,virus/worm infection and botnets; and 2) distribution and stability analysis of the popular port numbers and those of the popular source IP subnets in scans. We further propose several schemes to distribute the alarms more evenly across the SFCs, and improve the resiliency against the failures or attacks. 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. It significantly outperforms the traditional hierarchical approach when facing large amounts of *diverse* intrusion alerts.

Keywords

Distributed intrusion detection systems, Alert fusion, Load balancing, Distributed hash tables, Scalability.

1. INTRODUCTION

Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network/service operators. 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. However, many of these systems still suffer from high false positive and false negative rates due to the quick emergence of new attacks and the limited view of the local IDS. For instance, single point detection is often insufficient to determine the presence of a worm operating in the wild [32].

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 [27, 20, 12, 19, 35]. 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.

Such DIDS can belong to a single security service provider. For example, Symantec gathers security events from partner devices around the world with more than 20,000 sensors monitored in 180 countries.Alternatively, multiple ISPs can have the following incentives to share a cooperative DIDS system.

- Today's fast propagation of viruses/worms (*e.g.*, Sapphire worm) can infect most of the vulnerable machines in the Internet within ten minutesor even less than 30 seconds with some highly virulent techniques [29]. Thus it is crucial to identify such outbreaks in their early phases by integrating alerts from multiple sources.
- Such distributed monitoring is robust to various scan techniques that worm propagations may employ.
- Alert integration can help locate the attacker and zombies of spoofed denial-of-service (DOS) attack.

Furthermore, the ISPs can anonymize part of the symptom reports without affecting the alert fusion. In fact, recent worm modeling research has noted that distributed worm monitoring is much more effective than a centralized one. For example, a distributed monitor can detect the worm four times faster than a centralized monitor with the same size of monitored IP address space [21].

Nowadays, there are huge numbers of diverse alerts issued from the distributed IDS sensors. There are over 15 million intrusion alerts reported to DShield, the largest Internet firewall log repository, everyday. These alerts consist of many diverse variants. The recent Symantec Internet security threat report highlights a sharp rise in the number of Windows Virus/Worm variants: more than 7,360 new variants were documented from July 1 to Dec. 31, 2004, representing a 332% increase over the same period last year. As of Dec. 31, 2004, the total number of documented Win32 threats and their variants was approaching 17,500 [31]. In addition, there are over 18 thousands vulnerabilities found, according to CERT [5].

Such a trend demands the following features for a scalable DIDS infrastructure: 1) efficient routing of alarms related to different intrusion events to different SFCs without consulting any central directory server or flooding mechanisms among peering points. This is particularly important because although the existing hierarchical

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.
Copyright 2006 ACM 1-59593-417-0/06/0009 ...$5.00.

approach can effectively aggregate the attacks of the same type, it still encounters severe scalability problem when facing many different types of attacks simultaneously. 2) distributed queries support over multiple SFCs to aggregate information at various levels, *e.g.*, IP address prefixes, port number ranges; 3) load balancing; 4) robustness and fault-tolerance; and 5) attack resiliency.

Existing DIDSs fall in three categories: centralized/hierarchical fusion, publish/subscribe model, and P2P querying. But none of them can scale to the large number and various types of alerts as discussed in Section 2.

To address these challenges, we propose to build a distributed IDS fusion system based on emerging distributed hash table (DHT) technologies [22, 30]. A recent DARPA research agenda calls for building a Cyber Disease Control Center (CDCC) to defend against worms [32]. 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 SFC for a global view on the prevalence, cause, and patterns of such attacks. To the best of our knowledge, we are among the first to apply DHT to route various types of alarms for DIDS.

DHT provides some very nice properties as a decentralized routing and location infrastructure, such as scalability, fault-tolerance, robustness [22, 30]. However, the following challenges remain for building the CDDHT.

- How to design the routing key so that all related alarms are fused to the same SFC, while reducing other irrelevant alarms?
- Besides delegating fusion jobs to different SFCs, load balancing is very crucial for scalability. How to achieve load balancing among SFCs?
- How resilient is CDDHT against failures/attacks?

In this paper to address these questions, we make the following contributions.

- We design the routing key for four common types of intrusions: *DoS attacks*, *port scanning*, *virus/worm propagation* and *botnets*, based on their essential characteristics.
- With one-month of DShield firewall log data consisting of over 600 million scan records from over 2200 worldwide providers, we study the popularity dynamics of top port numbers and top source IP subnets of scans, and leverage it to design routing keys with good load balancing characteristics.
- We propose several other dynamic load balancing schemes, such as *load-aware node bootstrapping*, *node migration*, *virtual nodes*, *IDS alarm rate limitation* and *aggregation tree* to distribute the alarms more evenly across the SFCs.
- We show that CDDHT is much more resilient against failures/attacks than the traditional hierarchial DIDS, and propose some preliminary schemes to further enhance the attack resiliency of CDDHT.

We build the CDDHT system on top of Chord [30], and evaluate it by simulation with daily DShield firewall scan logs. The results show that we can effectively fuse the related alarms while distributing unrelated reports evenly among the SFCs. Due to the diverse alerts, CDDHT achieves much better balanced load when compared with the hierarchical approach.

2. RELATED WORK

Existing work on DIDS alert routing can be classified into three categories: hierarchical [27, 20], publish/subscribe models [12, 19, 35], and P2P querying [7, 4, 15].

Centralized/Hierarchical Model In the hierarchical model, the IDS sensors form a hierarchical structure, where high-level sensors receive the alerts from lower-level sensors [27, 20].

However, there are several drawbacks about this approach in terms of scalability and reliability. First, this approach is good to alleviate the load of root node by doing aggregation at each level when there is a small number of intrusion types. However, given that the number of different viruses/worms (different types of alerts) is also increasing at exponential speed, many of the alerts still have to be sent to the root node, which can be easily overloaded. The higher level the node is, the more load it tends to receive and thus more likely to become heavily loaded and crash. We also compare the hierarchical approach with CDDHT in Section 6. Second, the failure of any non-leaf node will isolate all the nodes in its subtree from the rest[1]. Third, the hierarchical tree structure faces the natural instability of a distributed network. The cost is high for a node joining/leaving the tree.

Publish/subscribe Model DIDS based on the publish/subscribe model include Indra [12], COSSACK [19], and DOMINO [35]. Basically, users sign on to different groups based on their interests, and then they share the information within a group with application-level multicast. For example, DOMINO is a hybrid of hierarchical and publish/subscribe models [35]. It applies hierarchical model in each domain, and then uses publish/subscribe for sharing among domain roots.

Though working well for small or moderate scale DIDS, the publish/subscribe model suffers $O({n_i}^2)$ communication where n_i is the number of nodes in group i, if every member in the group is attacked, *i.e.*, they would thus send alerts to everyone else. Given todays global-scale attacks, a publish/subscribe group for an emerging attack may have hundreds of thousands of subscribers or even more. In contrast, in our scheme, each node only sends alert to one SFC. After fusion, that SFC sends the fusion report to all the nodes which have sent related alerts in the past. Thus, the total communication is only $2 \times n_i$.

There are also some centralized publish/subscribe systems where similarly a single node collects all published information before disseminating them. However, it is hard to tell which node is responsible for which type of alerts, especially for unknown attacks. If all alerts go to one collector or need to lookup a directory server first, it becomes a centralized model as discussed before.

P2P Querying Recently, Netbait [7] is proposed as a P2P IDS system on top of PIER [10], a DHT-based distributed rational database which supports SQL-like queries. In Netbait, every node maintains its local worm alarm database, and use the PIER-supported distributed query to collect the desired information from all other nodes. However, to understand the global-scale attacks like viruses/worm propagation, Netbait has to query *every* node, and suffers the scalability problem. In contrast, our CDDHT system proactively fuse the alarms to a small set of SFCs. Thus, for query over any attack, the request only needs to be routed to a few SFCs. WormShield leverages DHT to collaboratively generate worm signatures [4]. By re-implementing the EarlyBird [26] in a distributed manner, they use the content block fingerprint to find popular contents, and then use address dispersion to identify worms. However, such signatures may not be accurate and cannot work against even simple polymorhphic worms. And their approach is very specific to the EarlyBird algorithms. In contrast, our CDDHT system is general and can work with any worm signature generation approach.

More recently, Locasto et al.developed an interesting approach to extract relevant information from alert streams and encode it

[1]This happens when each node only have single parent in the tree. A multi-parent tree can be more resilient, but is obviously a tradeoff to the complexity.

in Bloom Filters [15], which is complementary to our CDDHT scheme.

3. CDDHT SYSTEM ARCHITECTURE

The architecture of the CDDHT system is shown in Figure 1. There are two types of nodes in the DIDS system: multiple heterogeneous IDS sensors and SFCs. One strategy for selecting an SFC is to choose a subset of IDSs, who have more resources (*e.g.*, more CPU and bandwidth) and better security measures as the SFCs. Alternatively we can use some dedicated hosts to be the SFCs. In the CDDHT system, we use some pre-defined rules to recognize SFCs, *e.g.*, we can use the first bit of the node ID, "1" for SFCs, and "0" for normal IDSs.

All DHT systems provide two basic interfaces: 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`. DHT systems can guarantee a deterministic routing in $O(\log n)$ hops for a DIDS system of n nodes. This implies that the DIDS built on top of the DHT can be scalable to very large networks. Recent research also made DHT resilient to the node churn problem [24].

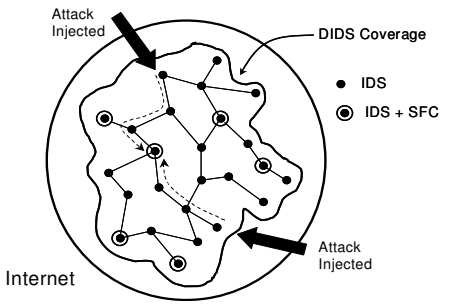


Figure 1: Architecture of the CDDHT system.

In CDDHT, when an attack is launched, each IDS attempts to locally detect the attack. Upon detection, it generates a symptom report that will be forwarded to the appropriate SFC with the `put(key,object)` operation. Thus, `object` is the attack symptom report, and `key` is the the routing ID which is generated from the report, and deterministically maps to a node on the DHT. Since the key we used is to route different types of attacks to different SFCs, we term the `key` as *disease key*. Similarly, for an attack with a disease key, every IDS in the CDDHT can query its prevalence with the `get(key)` operation. The node ID of an SFC is the concatenation of the SFC identifier (the first bit "1") plus the disease key, while the node ID of a IDS can be the concatenation of the IDS identifier (the first bit "0") plus some random hash (*e.g.*, SHA-2) of their IP addresses.

For instance, as shown in Figure 1, there is an attack detected by two IDSs, and both send alarms to the same SFC. The SFC will fuse alarms together to better characterize the attack. Finally the SFC can send the result back to the IDSs which have contributed to the detected attack scenario.

Based on the deterministic nature of the disease key routing, we can route the same types of attack reports to the same SFC. Transmission of the alerts will naturally form a tree rooted at the SFC. Though we have schemes in Section 5.1.3 to avoid congestion, the main advantage of CDDHT is to fuse vast number of *different* attacks because alerts will be distributed to many different SFCs with each responsible for different types of attacks. Thus CDDHT is highly scalable to large number of different attacks.

Instead of confining ourselves to specific algorithms for each types of attacks, the CDDHT system is proposed as a general framework which allows to plug in any fusion algorithm in any SFC. For example, for worm detection, the *Kalman filter* based worm trend detection [37] can be a good candidate. CDDHT system also can help collect suspicious worm samples for worm signature generation.

4. DISEASE KEY DESIGN

As events are generated locally at each IDS node, the attack symptoms must be reported and routed to one SFC which can then perform data fusion and inferences about such an attack. In this section, we design the routing keys to construct the CDDHT.

The challenge is: for a single intrusion, given the vast, diverse symptoms perceived from many heterogeneous IDSs around the world, how is the key effectively designed to fuse these events to a single SFC? For instance, while a worm is propagating, a router-based IDS may see much larger ranges of source/destination IPs 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 at either ingress, egress, or both.

Such key should also immune from attack pattern changes, like the number of zombies, the speed of attacks, the scope of victims, *etc.*. Furthermore, the design should strive to achieve load balancing while ensuring aggregation of related symptoms. All of these properties must be accomplished through key generation by each peer IDS in a decentralized and deterministic manner.

Given the requirements above, to ensure that related event symptoms will be routed to the same SFCs, we only use the intrinsic characteristics of each type of attacks, *i.e.*, the information that uniquely identifies related events, for the fields of the routing ID. Extraneous identifiers that may have clarified routing in some cases but confuse in others are thrown out. Since worms, DoS, botnets and port scans are the largest percentage of large scale attacks on the Internet, we focus on these four categories.

The disease key design is based on the following sources: 1) survey of various attacks: worms [28, 33], denial of service attacks [16, 17, 14] and port scans [13, 18]; and 2) study of the Internet intrusion characterization based on one-month DShield [25] data of over 600 million scan records.The format of disease keys is shown in Figure 2. The first element of the routing ID is a 3 bit identifier which can differentiate 8 types of intrusions for extensibility. Currently we only use 4 of them as discussed below.

4.1 DoS Attacks

Ideally, the disease key for DoS attacks should properly distinguish and correlate millions of DoS attack events, but DoS attacks are very hard to characterize.There may be one or many attack agents sending attack traffic, and these agents often have spoofed IP addresses, or may even be servers sending legitimate responses (*e.g.*, distributed reflection attacks). Other than application-specific DoS attacks (*e.g.*, DNS request attack), DoS attacks do not have a fixed port number.

For most of DoS attacks which attack one targeted server or network, the only invariant 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 IDS, and the "victim" IP address can be source or destination depending on the situation. Take a TCP SYN flood attack for example. For the IDSs

Intrusion	ID	Characterization Field(s)			Original Length
DoS Attack	000	Victim IP (subnet)			35 bits
Scans	001	0 (for vertical & block scan)	Source IP address		36 bits
		1 (for horizontal & coordinated scan)	Scan port number	Source IP (for horizontal scan)	52 bits
				0 (for coordinated scan)	
Viruses/Worms	010	0 (for known virus/worm)		Worm ID (32bit)	36 bits
		1 (for unknown virus/worm)		Destination port number	20 bits
Botnets	011	00 (for DDNS entry)		Botnet ID (32bit)	37 bits
		01 (for URL entry)		Botnet ID (32bit)	37 bits

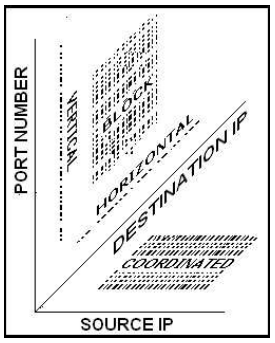


Figure 2: Disease Key of CDDHT (left) and the visual representation for four types of scans (right)

closest to the attack agent or the victim machine, the victim is the destination IP address. In IP Spoofing cases, while for the third party IDSs which receives "backscatter" traffic [17], the victim is the source IP address. In this way, we can correlate the alerts from multiple sources to track the zombies, even when their IP addresses are spoofed.

4.2 Port Scan

Given the extreme fast propagation of worms nowadays, to monitor and detect port scans in a timely manner is of crucial importance for early detection and prevention of worm outbreak. Based on source/dest IP and the port number involved in the scans, there are four well known types of scans: *horizontal scan, vertical scan, coordinated scan*, and *block scan* as designated in Figure 2.

Horizontal port scans are the most common type of port scans. Horizontal port scanning involves scanning a range of IP addresses for hosts listening on a specific port. We indicate horizontal port scans in our disease key with an identifier bit "1" and include the scan destination port number. The port number is unique because it reflects the vulnerability the virus/worm or attackers try to exploit. Unlike DoS attacks, the attacker needs to use a real source IP address, since he/she needs to see the result of the scan in order to know what ports are actually open[2]. Thus, we use the source IP address as the last field of the scan disease key. We can find the amount of scans for each source IP. Moreover, with range queries described in Section 5.4 we can aggregate some of the horizontal scans to coordinated scans, and get a more comprehensive view of attacks. In addition, we also can aggregate solely by port number, to get the most popular scanning port.

Coordinated scans can be viewed as horizontal scans from multiple sources.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.

A vertical scan is a scan of some or 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 exploits to attempt.We represent this scan with the identifier bit 0, and the source IP in the disease key. We do not include the port set because such a set is not stable and is hard to represent accurately within the small space in the disease key.

The fourth type of scan, a block scan, is a combination of horizontal and vertical scans over numerous services on numerous hosts. This scan can be regarded as a vertical scan without a fixed destination. We use the same disease key as with vertical scans. More complicated scans than these four are possible, but we have not seen much in practice, and for now we leave them as subsets of these major classifications.

[2]Although the 'idle scan" allows completely blind scans, it is based on predictable IP-ID sequence numbers. But recent versions of operating systems, such as OpenBSD, Solaris and Linux, have made the IP-ID sequences less predictable and rendered the attack obsolete [11].

4.3 Viruses/Worms

For existing ones, most IDSs use signature based approaches to detect them. Since their standardized names can be unique to distinguish them [2], we use an SHA-2 hash of the names of viruses/worms as the disease key. Unknown viruses/worms can be detected through scanning, or approaches like DSC [9]. But given any virus/worm, the intrusion (destination) port number is normally unique, and we use that as the disease key. We also consider other alternatives. For example, the worm signature in the form of bit patterns may not be unique due to the polymorphism. Recently, there emerge many semantics based signature to deal with such problem, such as the Turing test signature which captures the inherent vulnerability that a worm explores [3]. It is our future work to design a standard format for such signatures and include it as part of the disease key.

4.4 Botnets

A botnet is a network of compromised machines running programs (usually referred to as *bot*, *zombie*, or *drone*) under a common *Command and Control* (C&C) infrastructure [8]. Botnets are frequently used as underlying facilities to recruit zombies for performing DoS attacks [8]. Usually, when a machine is compromised, it is injected with a bot program. Then the bot will try to connect to a hard-coded C&C server via IRC or HTTP to receive commands from the hacker. To make the control over the botnet more resilient, for IRC server based control, most hackers choose to hard-code the dynamic DNS name into their code so that they can change the C&C server by changing the DNS entry when needed [8]. Another popular way to control a botnet is to put the commands on a public web site and include the URL into the bot code. This can better hide botnet traffic as normal web traffic. In either way, the DNS entry or the URL is unique to distinguish different botnets.

We take two steps to generate the botnet ID for disease key. First, we use a 128-bit SHA-2 hash of the DNS entry or URL because the length of URLs vary and SHA-2 is good at hashing data of various length. Such 128-bit SHA-2 digest is then fed to a 4-universal hash function to get the 32 bit botnet ID. In addition, we use 2 bits to represent the different types of botnets, *e.g.*, 00 for the DDNS entry (for the method other than URL) and 01 for the URL entry.

These carefully chosen characteristics form the *heterogenous* dimensions for the disease key. As shown in Figure 2, the keys have different length for different symptoms: viruses/worms may only need 20 bits, while the keys for horizontal scans can have up to 52 bits. For most DHT systems [30, 22] given an n-node DHT, in an even node distribution, the effective key length is $\log n$. Then even a 10^6 node DHT can only have 20-bit effective keys. We can in-

crease 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 5.4, we cannot hash the key randomly to distribute the load. A large key space can easily lead to unbalanced load distribution. In following sections,we will address these problems.

5. FEATURES OF CDDHT

In this section, we discuss load balancing, and failure and attack resilience of CDDHT.

5.1 Load Balancing

Internet attacks are increasing in frequency, severity and sophistication. When a global attack occurs, many IDSs will detect it and send alerts, which may easily overload some SFC(s). Furthermore, some characterization fields in the disease key have strongly uneven popularity distribution, like port numbers 80 and 135 for scans. We use as much essential characteristics of attacks as possible for the disease key to spread the alert loads among the SFCs. In addition, we design the following techniques.

5.1.1 Proactive Balancing with Stable Hot Spots

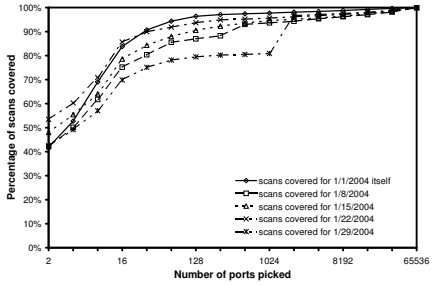


Figure 3: Scan coverage stability of the most popular ports chosen at 1/1/2004

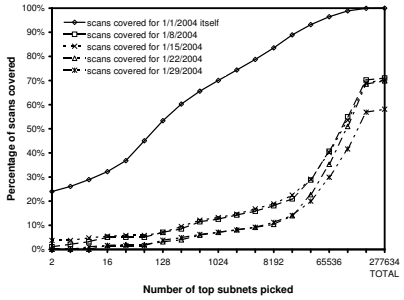


Figure 4: Scan coverage stability of the most popular subnets of 1/1/2004

Since the number of SFCs is much less than the size of disease key space, for the characterization fields with hot spots, we want to design the disease key so that each hot spot gets at least one dedicated SFC because we have to somehow fuse all related alerts to one SFC to obtain a global view. There are more mechanisms to further alleviate the load of SFC as we will describe later.

However, this scheme is only useful when hot spots are stable. Since port scanning is the most common type of Internet event alarm, we study the popularity distribution and its stability of the characterization fields as an example. Previous work suggests that the popularity of scan port number and source IP address follows heavy tail distribution, *i.e.*, a small number of entities are responsible for a large number of scans [34]. They also found that popular source IP can only remain hot within hours, but not even a day [35]. However, it remains unclear how stable the popular scan port numbers and the source IP prefixes are. The latter is important because as discussed before, the key space of DHT is much smaller than that of the disease key. Thus only the prefixes of the IP address are indeed used to route the alarm for fusion.

To this end, we study the distribution stability of the popular scanned ports, and those of top source IP subnets, based on the stability of the scan record coverage from previous popular entities, *e.g.*, how many scan records from today are covered by the top 10 ports of 10 days before. Our analysis is based on one-month (January 2004) of intrusion logs from DShield [25] consisting of over 600 million scan records from over 2200 providers around the world.

As shown in Figure 3, 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. We further study the stability of the class C (/24) subnets of source IP as shown in Figure 4. Unlike port numbers, the popular source IP subnets of scans keep changing. That is, there is no steady blacklist. See [6] for more results and analysis.

Leveraging stable ports for load balancing depends on the type of underlying DHT. In Chord, we use 7 bits for the port number with 128 buckets. We map each of the 64 popular ports into one unique bucket, and map the other ports randomly into the remaining buckets. This effectively improves the load balancing especially when the SFC nodes are relatively few and thus sparse in the node ID space.

5.1.2 Balancing Load of the Key Space

Since each SFC s is on the DHT routing paths to many other SFCs, it can continuously collect the load information of other SFCs (when there is an alert sending through s) as well as the load of its neighbors. If s notices some SFC overloaded, it can migrate to that region for load sharing under the following conditions: 1) the load of s has been below a certain threshold for a certain period; 2) s can dump its load to its neighbors without overloading them. Then s can leave, and rejoin the system using migration.

Alternatively, an SFC with low load can spawn a virtual node on itself and have the virtual node join the CDDHT through the "load-aware bootstraping" which happens only when we add an IDS node or a new SFC node as an SFC. The new SFC first finds a bootstrap node b in the CDDHT with approaches described in [22, 36], and queries b for the SFC node with the heaviest load from the locally maintained load table of b. It will then join and split the load with the overloaded SFC.

5.1.3 Balancing Load of Single Hot Key

IDS Alarm Rate Limiting A router IDS may generate many alarms for one specific attack. We can set some alarm rates for the IDS so that it executes some temporal aggregation to combine several alarms within one time interval to a single alarm.

Aggregation Tree for Large-Scale Attacks When a global-scale attack occurs, the number of alerts sent to an SFC may be overwhelming because all these alerts may have the same unique disease key. For instance, in an intensive DoS attack or a large-scale worm outbreak, all the keys are the same and thus all alerts will be sent to a single SFC. To address this issue, we combine the hierarchical approach with CDDHT – the IDSs and SFCs on the routing path can fuse the alerts locally before passing it on when it observes a big burst of alerts to the same destination. This will be only triggered by very large burst of alerts.

Actually, by using this technique, the alarms will follow an aggregation tree towards the final destination SFC. Each node in the tree has $O(\log n)$ neighbors, where n is the total number of nodes in the CDDHT system. Thus for any IDS/SFC, the amount of

alarms received per aggregation time interval (*e.g.*, 10 seconds) for each attack is bounded by $O(\log n)$.

5.2 Failure Resilience

Each DHT node periodically sends updated neighbor table to its neighbors. When some node fails, its neighbors will detect it, and update their routing tables to bypass the failed node. If a SFC s fails, the SFC which has closest DHT node ID to s will automatically takes over and receives the alerts. We further have each SFC periodically send their recent fusion results to a few SFCs which have close node IDs. Thus failure of a small portion of nodes has little impact on the CDDHT.

5.3 Attack Resilience

In this section, we first compare the DoS attack resilience of CDDHT against hierarchial DIDS; then propose some mechanisms to make the CDDHT more resilient.

5.3.1 DoS Resilience Comparison with Hierarchical Model

We consider two scenarios, with or without connectivity/topology knowledge, for comparison. In each scenario, we assume that attackers can only attack one (or a small number of) node in the system.

With the DIDS connectivity knowledge, it is straight forward to attack the root node of the tree, or all the nodes of a high level in the tree (*e.g.*, level 2 or 3, so the total number is still small). Such attack can render all other nodes to be disconnected from SFC.

However, in CDDHT, even with the full connectivity knowledge, the best target is just to attack certain SFC. However, all other SFCs still function well. Furthermore, the functionality of attacked SFC will be shifted to other nodes with the load balancing scheme in Section 5.1.2.

In the case that the attacker is not topology aware we want to compare the average case performance of hierarchical DIDS and that of the CDDHT. More specifically, assume that the attacker bring down a random node, we compute the average number of nodes get disconnected, *i.e.*, the number of nodes which cannot connect to the root SFC in the hierarchical DIDS or the number of nodes which cannot connect to the corresponding SFCs for the alerts they have in CDDHT.

Consider a perfect k-way tree network, the loss of one node will result in one (itself) or more (children) nodes disconnected from the root. When taking all nodes into account, we have the following theorem.

THEOREM 1. *The average number of nodes disconnected, given one node loss in a k-way hierarchical DIDS, is $O(\log n)$ where n is total number of nodes.*

PROOF. A loss at the top level (root) will result in n nodes being lost. Then at a depth of x from the top there are k^x nodes, each loss case at depth of x result in $m = \frac{n-\sum_{y=0}^{x-1} k^y}{k^x} = \frac{n-\frac{k^x-1}{k-1}}{k^x}$ disconnected nodes. Therefore, the sum of all loss cases at a given depth of the tree is $k^x \times \frac{n-\frac{k^x-1}{k-1}}{k^x} = n - \frac{k^x-1}{k-1}$, and a perfect k-way tree has a depth of l in $O(\log n)$. The sum of all loss cases is $ln + \frac{n-l}{k-1}$, giving a loss case mean of $(l - \frac{1}{k-1}) + \frac{l}{(k-1)n} \approx l$ disconnected nodes, which is $O(\log n)$. □

In contrast, in CDDHT, each node has $O(\log n)$ neighbors, and any one of them fail will not affect the report delivery unless the failure node is the destination itself. When it happens, the following theorem shows that CDDHT has better resilience even when ignoring the self-healing of CDDHT.

THEOREM 2. *The average number of nodes disconnected, given one node loss in CDDHT, is $O(1)$.*

PROOF. Suppose there are m SFCs out of a total of n nodes in CDDHT, the probability of having a SFC failure is $\frac{m}{n}$. When a node has a report, there is a probability of $\frac{1}{m}$ that it should be sent to the failed SFC. Thus the total average number of disconnected nodes is $O(\frac{m}{n} \times \frac{1}{m} \times n) = O(1)$. □

5.3.2 Authenticity of Alarms

To enforce the authenticity of alarms, when each IDS joins the CDDHT, we will validate its request by querying *whois* servers and checking the BGP tables from multiple vantage points. The purpose is to check whether the origin of the IDS is correct, and whether the range of the IP addresses it covers is the same as it claims. Furthermore, we may contact the administrator of that domain to further validate the authenticity of the request. With such validation, when an IDS reports attacks, we can check whether the involved IP addresses belong to the domain of the IDS.

Moreover, when every node (including both SFC and IDS) joins CDDHT, we will generate their public-private key pair based on DSA [1] algorithms. The trusted certification authority (CA) of the CDDHT will generate the certificate for the public key of each node. When an IDS d sends a symptom report, it appends its digital signature and its public key certificate to the symptom report. The receiver SFC will check d's public key with the CA, and then uses it to verify the signature of the report. When a IDS/SFC on the path needs to do aggregation when it receives bursty alarms that need to be forwarded, it will sign such changes and append its own public key certificate to the aggregated report.

With such mechanisms, unless an IDS or SFC is compromised, we can always detect the bogus alarms.

5.3.3 Dealing with Compromised Nodes

Sometimes an IDS can be compromised by attackers. They can use these nodes to send bogus symptom reports or make up aggregated reports for the alarms routed through the CDDHT system. To minimize their influence to the overall CDDHT system, we evaluate the severity of one global attack, not only by the total number of alarms, but also by the range of IP addresses influenced by the attack, and the number of IDSs reporting that attack. For aggregated reports, the SFC which is responsible for the attack will randomly check some IDSs which have attack reports aggregated. Each IDS/SFC will keep the original alarm reports received (from which the aggregation reports are generated) for certain period of time. Thus once cheating is detected, we can trace back to find the compromised node. Then we can update the routing table of all its neighbors to remove it from CDDHT. This is much easier to fix than that of the hierarchical based approach.

When a SFC is compromised, it is a more severe problem because we cannot obtain correct results from it before it is removed from the CDDHT system. But usually a SFC is much better secured than an IDS. And in contrast to hierarchical approaches, each SFC is only responsible for a small set of attacks, the remaining CDDHT system still function well.

We apply the "trust but verify" principle to detect compromised SFCs. We envision that there is a centralized authority which periodically checks the fusion results from each SFC. When disseminating the fusion results, each SFC also needs to send results to the authority along with a list of IDSs which reported related alerts and the number of such alerts from each IDS. The authority will randomly choose a subset of such IDSs for verification. If the result is inconsistent, the authority will verify with more IDSs. If many IDSs report inconsistency, most likely the SFC is compromised. Otherwise, the authority check other SFCs for any reports from the

inconsistent IDSs to see if these IDSs often cause inconsistency. If so, most likely the IDSs are compromised.

5.4 Querying on CDDHT

CDDHT can support two types of queries for fusion results, queries with a given disease key or *aggregate queries*. The former is easy, simply call `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 a collector, and query all other nodes in the "zone" which have the same node ID as the collector except the last few bits for source IP addresses. We also plan to leverage "range queries" over DHT [23] for more complicated query requests.

6. PRELIMINARY EVALUATION

6.1 Methodology

We implemented a preliminary CDDHT system based on the Chord [30] simulator with proactive balancing with stable hot ports and virtual nodes, and simulated the scan alert fusion with DShield firewall logs. The data only contain scan records, which arguably represent the largest number of intrusions in the Internet. It is our future work to collect data on other attacks for a more comprehensive evaluation of the system.

To compare with the hierarchical approach, and to examine closely at the instantaneous load distribution, we also implemented an event-driven simulator where each alert is treated as an event that is originated from certain IDS based on its timestamp. Such event is forwarded to its parent or to a certain SFC through CDDHT routing. For both hierarchical and CDDHT approaches, each SFC fuses the alerts with the same disease key at the end of every minute interval. For hierarchical DIDS, each SFC passes the fused alert to its parent, which will be fused at the next time interval. For CDDHT, we assume that the symptom report transmission delay is much smaller than the time interval and is ignored.

We evaluated our work using one-month daily DShield data, and found that the results from each day are similar. For this reason, we will show the results for that of January 2nd, 2004. It contains over 25 million scan logs from 1417 providers. These scan logs are classified into the events in Table 1.

The CDDHT consists of 2,200 nodes as the number of providers from one-month DShield data. We randomly choose 10% of them as SFC. Similarly, we set the fanout of each node in the hierarchical tree to be 10 so that the number of non-leaf nodes (which work as SFC to fuse alerts) is about 10% of the total number of IDS nodes. In fact, we found that the load balancing results for the hierarchical approach are not sensitive to the fanout of the tree. We then used these classified scan events to generate the disease keys. See [6] for the details on simulation set up.

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

Table 1: Number of scans and their types in the attacks

The metrics to evaluate our work include the effectiveness of fusion, and the variation of load among the SFCs. It is our future work to evaluate the failure/attack resilience of CDDHT. The fusion effectiveness is the percentage of related alarms (defined by the disease key) fused in the corresponding SFC. Because Chord has deterministic routing, if given the same disease key, it always routes to the same SFC node. Therefore, we can get 100% effectiveness of fusion in CDDHT. For each SFC, the load is denoted

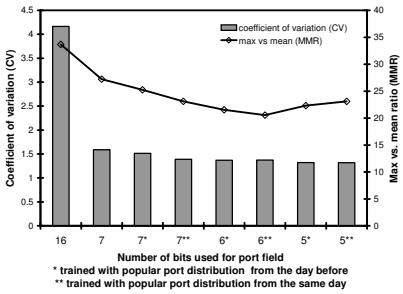


Figure 5: Load balancing results with stable hot ports.

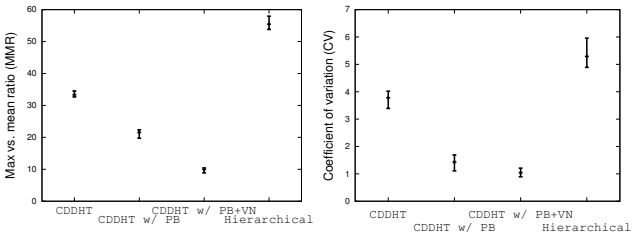


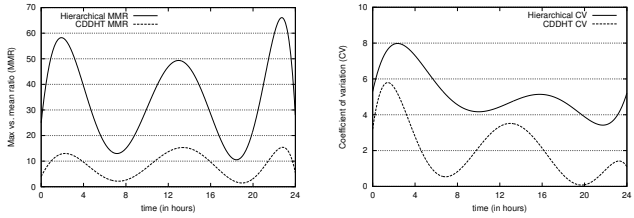
Figure 6: Load variation comparison between hierarchial scheme and CDDHT: MMR (left) and CV (right). PB means proactive balancing with stable hot ports. VN stands for virtual nodes.

by the number of symptom reports received. The load variation is measured in terms of the coefficient of variation, $CV(x) = \frac{standard\ deviation(x)}{mean(x)}$, and the maximum vs. mean ratio (MMR), as shown in Figure 5. The CV 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.

6.2 Results

The stability study in Section 5.1.1 suggests we map the top 64 most popular ports to the half buckets of 7 bits of port evenly, and hash other ports to the remaining half. In practice, we found that it can significantly improve the load balancing: it reduces the CV by more than 60%, and reduce the MMR by about 40%, compared with using the entire 16 bit port number, as shown in Figure 5. There are still certain hot spots because some scan events are much more popular than the rest, as indicated in the MMR ratio.

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. See [6] for more detailed results.



MMR CV

Figure 7: Dynamics of load variation simulated with data of 1/2/2004.

We further applied virtual nodes to alleviate the hot spots. The effect of node migration is similar. As described in Section 5.1.2, each SFC computes the average load based on the load information piggybacked with the alerts. If its load is more than four times the average for more than 10 minutes, it will choose the SFC with the

smallest load and spawn a virtual node on it. Such virtual node will be released with the load is less than the average for more than 10 minutes. For each setting, we ran the experiment for 10 times and show the median, 10 percentile and 90 percentile in Figure 6. The virtual nodes further reduce the MMR by about 50% and reduce the CV by 30%. Compared with the hierarchical approach, we reduced the MMR by a factor of 5.5, and reduce the CV by a factor of 5.2. Note that aggregation tree is not applied to CDDHT yet.

The dynamics of load variation is shown in Figure 7, where the three peaks in MMR correspond to the three peak time intervals when large amount of scans are received. The MMR for CDDHT is much smaller and smoother than those of hierarchical approach. The difference for CV is less dramatic, but has similar trend. Note that the CDDHT system gradually adds virtual nodes to evenly distribute the load, so the CV is getting smaller.

7. CONCLUSION AND FUTURE WORK

In this paper, we propose to build a distributed IDS based on distributed hash table (DHT). We embed the intrusion symptoms into the DHT dimensions so that alarms related to the same intrusion will be routed to the same SFC with good load balancing. For future work, we will evaluate other load balancing schemes we designed and investigate further on the attack-resilience and querying of CDDHT.

8. REFERENCES

[1] *DIGITAL SIGNATURE STANDARD (DSS)*, 1994. FIPS PUB 186.

[2] Virus names could be standardized, 2004. Computer Business Review Online, `http://www.cbronline.com/article_news.asp?guid=11D11704-DE5B-45BD-AF4B-%45D8F44E055C`.

[3] BRUMLEY, D., NEWSOME, J., SONG, D., WANG, H., AND JHA, S. Towards automatic generation of vulnerability signatures. In *Proc. of the IEEE Symposium on Security and Privacy* (2006).

[4] CAI, M., ET AL. Collaborative internet worm containment. In *IEEE Security and Privacy Magazine* (May/June 2005).

[5] CERT CC. Statistics 1988-2005. `http://www.cert.org/stats/cert_stats.html`.

[6] CHEN, Y., BEACH, A., AND SKICEWICZ, J. Cyber disease monitoring with distributed hash tables: A global peer-to-peer intrusion detection system. Tech. Rep. NWU-CS-04-040, Northwestern University, 2004.

[7] CHUN, B. N., ET AL. Netbait: a distributed worm detection service. Tech. Rep. IRB-TR-03-033, Intel Research Berkeley, 2003.

[8] FREILING, F., HOLZ, T., ET AL. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. Tech. Rep. ISSN-0935-3232, RWTH Aachen, 2005.

[9] GU, G., ET AL. Worm detection, early warning and response based on local victim information. In *Proceedings of the Computer Security Applications Conference (ACSAC)* (2004).

[10] HUEBSCH, R., ET AL. The architecture of PIER: an internet-scale query processor. In *Conference on Innovative Data Systems Research (CIDR)* (2005).

[11] INSECURE.ORG. Idle scanning and related ipid games, 2004. `http://www.insecure.org/nmap/idlescan.html`.

[12] JANAKIRAMAN, R., ET AL. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *IEEE WETICE Workshop on Enterprise Security* (2003).

[13] JUNG, J., ET AL. Fast portscan detection using sequential hypothesis testing. In *IEEE Symposium on Security and Privacy* (2004).

[14] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In *WWW* (2002).

[15] LOCASTO, M. E., PAREKH, J., KEROMYTIS, A. D., AND STOLFO, S. J. Towards collaborative security and p2p intrusion detection. In *Proc. of the IEEE Information Assurance Workshop (IAW)* (2005).

[16] MIRKOVIC, J., AND REIHER, P. A taxonomy of DDoS attack and defense mechanisms. *ACM CCR 34* (2004).

[17] MOORE, D., ET AL. Inferring Internet denial-of-service activity. In *USENIX Security Symposium* (2001).

[18] MOORE, D., ET AL. Internet quarantine: Requirements for containing self-propagating code. In *IEEE Infocom* (2003).

[19] PAPADOPOULOS, C., ET AL. Coordinated suppression of simultaneous attacks. In *DARPA Information Survivability Conference and Exposition (DISCEX)* (2003).

[20] PORRAS, P. A., AND NEUMANN, P. G. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *the 20th NIS Security Conference* (1997).

[21] RAJAB, M., ET AL. On the effectiveness of distributed worm monitoring. In *Proc. of USENIX Security Symposium* (2005).

[22] RATNASAMY, S., ET AL. A scalable content-addressable network. In *ACM SIGCOMM* (2001).

[23] RATNASAMY, S., ET AL. Range queries over DHTs. Tech. Rep. IRB-TR-03-011, Intel Research, 2003.

[24] RHEA, S., GEELS, D., ROSCOE, T., AND KUBIATOWICZ, J. Handling churn in a dht. In *Proc. of USENIX* (2004).

[25] SANS INSTITUTE. Dshield.org: Distributed intrusion detection system. `http://www.dshield.org/`.

[26] SINGH, S., ESTAN, C., VARGHESE, G., AND SAVAGE, S. Automated worm fingerprinting. In *Proc. of the ACM OSDI* (2004).

[27] SNAPP, S. R., ET AL. DIDS (distributed intrusion detection system) - motivation, architecture and an early prototype. In *the 14th National Computer Security Conference* (1991).

[28] STANIFORD, S., ET AL. How to own the Internet in your spare time. In *the 11th USENIX Security Symposium* (2002).

[29] STANIFORD, S., ET AL. The top speed of flash worms. In *Proc. of ACM CCS WORM Workshop* (2004).

[30] STOICA, I., ET AL. Chord: A scalable peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM* (2001).

[31] SYMANTEC INC. Symantec internet security threat report, March 2005. `http://enterprisesecurity.symantec.com/content.cfm?articleid=1539`.

[32] WEAVER, N., ET AL. Large scale malicious code: A research agenda. Tech. rep., DARPA Sponsored Report, 2003.

[33] WEAVER, N., ET AL. A taxonomy of computer worms. In *the First Workshop on Rapid Malcode (WORM)* (2003).

[34] YEGNESWARAN, V., ET AL. Internet intrusions: Global characteristics and prevalence. In *ACM SIGMETRICS* (2003).

[35] YEGNESWARAN, V., ET AL. Global intrusion detecction in the DOMINO overlay system. In *Proc. of NDSS* (2004).

[36] ZHAO, B. Y., ET AL. Tapestry: A resilient global-scale overlay for service deployment. *IEEE JSAC 22*, 1 (2004).

[37] ZOU, C. C., ET AL. Monitoring and early warning for internet worms. In *Prof. of ACM CCS* (2003).