Improving Attack Aggregation Methods Using Distributed Hash Tables

Zoltán Czirkos, Márta Rencz, Gábor Hosszú
Department of Electron Devices
Budapest University of Technology and Economics
Budapest
{czirkos,rencz,hosszu}@eet.bme.hu

***Abstract*—Collaborative intrusion detection has several difficult subtasks to handle. Large amount of data generated by intrusion detection probes has to be handled to spot intrusions. Also, when correlating the pieces of evidence, the connection between them has to be revealed as well, as it may be the case that they are part of a complex, large-scale attack. In this article, we present a peer-to-peer network based intrusion detection system, which is able to handle the intrusion detection data efficiently while maintaining the accuracy of centralized approaches of correlation. The system is built on a distributed hash table, for which keys are assigned to each piece of intrusion data in a preprocessing step. This method allows one to make well-known correlation mechanisms work in a distributed environment.**

***Keywords*-collaborative intrusion detection; attack correlation; peer-to-peer; distributed hash table.**

I. Introduction

In the earliest days of the Internet, services on the network were all based on trust. As e-commerce emerged, network hosts became victim of a wide range of everyday attacks. Due to the high amount of confidental data and resources that can be exploited, the possibilites and open nature of the Internet opened serious security questions as well.

The attacks network administrators fight against are both human and software controlled. They get more and more sophisticated, originating or targetting ocassionally multiple hosts at the same time. A large number of nodes can be simultaneously scanned by attackers to find vulnerabilities. Automatized worm programs replicate themselves to spread malicious code to thousands of vulnerable systems, typically of home users. Others compromise hosts to build botnets, which can deliver millions of spam e-mails per day.

As the manifestation of attacks, e.g., the evidence that can be observed is spread across multiple hosts, these large-scale attacks are generally hard to detect accurately. To recognize such, one has to first collect or *aggregate* the evidence, then *correlate* the pieces of information collected. A collaborative intrusion detection system has to analyze the evidence from multiple detector *probes* located at different hosts, and even on different subnetworks. However, this poses several problems to solve:

- large quantities of possible evidence collected,
- including inadequate data for precise decision making,
- communication and reliability problems,
- frequent change of intrusion types and scenarios.

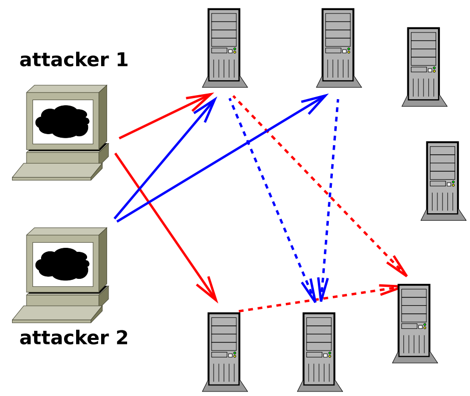

Figure 1. Messages carrying attack information in the Komondor system. If any probes in the network detect a suspicious event, it sends a report to the DHT. The nodes of the DHT act as correlation units as well, and are able to collect these reports.

Some of these troubles are specific for the isolated, host-based detection systems, while others occur only in case of the network scale intrusion detection. Despite of all these difficulties it is still worth collecting and correlating evidence available at different locations for the efficiency and accuracy boost of both detection and protection.

In this paper, we present a collaborative intrusion detection system, which organizes its participants to a *peer-to-peer (P2P) overlay network,* as seen in Figure 1. For intrusion data aggregation, a *distributed hash table* (DHT) is used, which is built on the Kademlia topology. This is used to balance the load of both aggregation and correlation of events amongst the participants. The organization of nodes in the overlay network is automatic. Should some nodes quit or their network links fail, the system will reorganize itself.

The rest of this paper is organized as follows. In Section II, we first review existing research of collaborative intrusion detection systems. Then we present the architecture of our intrusion detection solution based on the Kademlia DHT overlay in Section III. The results of our intrusion detection method and statistics of detection are highlighted in Section IV. Research is concluded in Section V.

 ISBN: 978-1-61208-201-1

II. Related Work

Attackers use various ways for intrusion of computer network systems depending on their particular goals. These methods leave different tracks and evidences, called the *manifestation of attacks* [1]. To discuss the internals of a collaborative intrusion detection system, we use the following terms [2]:

- *Suspicious events* are primary events, that can be detected at probes. Not necessarily attacks by themselves, but can be part of a complex attack scenario.
- *Attacks* are real intrusion attempts, which are used to gain access to a host or disturb its correct functioning. Usually these are made up from several suspicious events at once.

The activity of an SSH (Secure Shell, a remote login software) worm program can be seen as an example of an attack. These worms use brute-force login attempts using well-known user names and simple passwords [3], directed against a single host. The attempts are events that make up the attack in this case. Multiple failed login attempts usually indicate an attack, while a single failed attempt is usually only a user mistyping his password.

A. Centralized Collaborative Intrusion Detection

To achieve *collecting* and *correlating* events detected by a number of detector probes, various collaborative intrusion detection systems (CIDS) have been proposed, for which a detailed overview can be found in [4].

The earliest collaborative detection systems used a centralized approach for *collecting* the events. The Internet Storm Center *DShield* project collects firewall and intrusion detection logs from participants, uploaded either manually or automatically [5]. The log files are then analyzed centrally to create trend reports.

The *NSTAT* system [6] on the other hand is more advanced in the sense that its operation completely real-time. In NSTAT, the detection data generated by the probes is preprocessed and filtered before being sent to a central server for correlation. This system analyzes the order of events using a state transition mechanism with predefined scenarios to find out the connection between them.

The advantage of centralized methods is that the server is able to receive and process all data that could be gathered. Processing, i.e., correlation can be carried out with several different methods. SPICE [7] and CIDS [8] group events by their common attributes. The LAMBDA system tries to fit events detected into pre-defined and known scenarios [9]. The JIGSAW system maps prerequisites and consequences of events to find out their purposes [10].

B. Hierarchical and P2P Collaborative Intrusion Detection

By using hierarchical approaches, the scalability problem of centralized intrusion detection systems can be handled. The *DOMINO* system is used to detect virus and worm activity. It is built on an unstructured P2P network with participants grouped into three levels of hierarchy [11]. The nodes on the lowest level generate statistics hourly or daily, therefore they induce only a small network traffic.

The *PROMIS* protection system (and its precedessor, Netbiotic) uses the JXTA framework to build a partly centralized overlay network to share intrusion evidence [12]. The nodes of this system generate information for other participants about the frequency of detected suspicious events. This information is used to fine-tune the security settings of the operating system and the web browser of the nodes. This creates some level of protection against worms, but also decreases the usability of the system.

The *Indra* system is built on the assumption that attackers will try to compromise several hosts by exploiting the same software vulnerability [13]. If any attempts are detected by any participant of the Indra network, it alerts others of the possible danger. Participants can therefore enhance their protection against recognized attackers, rather than developing some form of general protection.

The scalability and single point of failure problems of centralized solutions can also be solved by using structured P2P application level networks. The P2P communication model enables one to reduce network load compared to the hierarchical networks presented above. The CIDS system [8] is a publish-subscribe application of the Chord overlay network. Nodes of this system store IP addresses of suspected attackers in a blacklist, and they subscribe in the network for notifications that are connected to these IPs. If the number of subscribers to a given IP address reaches a predefined threshold, they are alerted of the possible danger. The Chord network ensures that the messages generated in this application will be evenly distributed among the participants [14].

C. Structured P2P Networks

Structured P2P networks generally implement *distributed hash tables* [15]. DHTs store $\langle key; value \rangle$ pairs and allow the quick and reliable retrieval of any $value$ if the key associated to that is known precisely. This is achieved by using a *hash function* and mapping all data to be stored to the nodes selected by the distance of the hashed keys and their NodeIDs, which are chosen from the same address space. The connections between nodes are determined by their NodeID selected upon joining the network. They are selected so that the number of steps between any two node is usually in the order of $\log N$, where N is the count of all nodes.

DHTs all implement routing between their nodes in the application level to build the topology desired. The *Kademlia* network uses a binary tree topology [16], in which the distance is calculated using the XOR function. All nodes have some degree knowledge of the successively smaller subtrees of the network they are *not* part of. For any of these subtrees they have routing tables called k-buckets, which store IP addresses of nodes that reside in distant subtrees. When a node looks up a selected destination, it successively queries other nodes, which are step by step closer to the destination. The queried nodes answer by sending their k-buckets to the source. As nodes closer to the destination have greater knowledge of their neighbors, the lookup will get closer every step, as discussed in [16]. The distance in the XOR metric is halved with every message, so the number of messages is $\log_2 N$ with N being the number of nodes in the tree.

III. The Komondor System Architecture

In this section, our intrusion detection system named *Komondor* is presented. Its most important novelty is that it uses the Kademlia DHT to store intrusion data and to disseminate information about detected events. Having analyzed the collected events, Komondor correlation units may start an alert procedure notifying other nodes of the possible danger if necessary.

A. Distributing Load Among Multiple Correlation Units

The Komondor peer-to-peer application level network consists of multiple nodes. All nodes have the *responsibility* of collecting and correlating intrusion data. They also report attacks discovered to other nodes of the network, as seen in Figure 1. All participants of the Komondor network serve as intrusion detection units and correlation units as well.

The Komondor network is designed to enable the previously mentioned correlation methods to be used in a distributed manner:

- Pieces, which are correlated should be sent to the same correlation unit, so that it can gather all the information about the attack.
- Pieces of evidence, which are part of distinct ongoing attacks should preferably be sent to different correlation units. This reduces load and improves overall reliability of the system.

Komondor achieves this goal by *assigning keys to preprocessed intrusion data*, as seen in Figure 2. Keys assigned are used as storage keys in the DHT as well. For different attackers or attack scenarios, different keys are selected, and this way data is aggregated at different nodes of the Komondor overlay.

Correct key selection is critical, since pieces of evidence, which might be correlated to each other must be assigned the same key and sent to the same Komondor node for correlation. Note that these pieces do not have to be detected by the same probe, yet they can be aggregated by the same correlation unit. The Komondor system is essentially a *middle layer inserted into the intrusion detection data path.* The nodes of the DHT are the correlation units, which have to implement the same correlation methods as their centralized counterparts. However, the correlation procedure is started as soon as the preprocessing stage with the key selection, and it is finalized at the correlation units.

The detected and preprocessed data of suspicious events is stored in the Komondor overlay. In this system, the key assigned at the preprocessing stage of detection is used as a *key for DHT operations* as well. The value parts of the $\langle key; value \rangle$ pairs stored are any other data, which might be useful for detection or protection. As all nodes use the same key selection mechanism and the same hash function, events related to each other will be stored at the same node, as seen in Figure 1. This way the algorithm ensures that the aggregator node has perfect knowledge of all events related to the attack in question.

The reason why a structured overlay – Kademlia – was selected for the Komondor system is that it has the advantages of distributed and centralized detection systems as well. Event data collected has to be sent to a single collector node only (this would not be possible with an unstructured overlay, as those have no global rule to map a key to a node.) Moreover, when Komondor nodes are under multiple but unrelated attacks, the network and computational load of both aggregation and correlation is distributed among nodes. The Komondor system neither has a single point of failure: the responsibility of correlating particular events is transferred to another node in this case. The overlay can also be used to disseminate other type of information as well, for example the attack alerts, which enable nodes to create protection.

B. Kademlia as the DHT Topology of Komondor

The nodes of Komondor create a *Kademlia DHT* overlay. This is the topology, which can adapt its routing tables to the dynamic properties of traffic generated by the intrustion detection probes. As discussed below, other DHTs wouldn't be able to adapt their routing tables to the dynamic properties of this kind of traffic.

Storing information of events generated by the probes generates significant overlay traffic, which will load not only detector and collector nodes, but other nodes along the path from the former to the latter one as well, as routing between nodes is handled on the application level. If the *events are in correlation* with the same attack, the *key chosen is likely to be the same*, making the distribution of keys highly uneven. However, by using Kademlia, network traffic can be significantly reduced in this scenario. The reason for this is that the routing algorithm of Kademlia is very flexible: any node can be put to the routing tables of any other node while still obeying the rules of the routing protocol. Routing tables of other DHT overlays like CAN or Chord are much more rigid, and therefore the routing algorithm of those cannot optimize the number of messages for the store requests with the same key.

Table I compares the number of messages generated in intrusion detection for Kademlia and Chord, with the latter

 ISBN: 978-1-61208-201-1

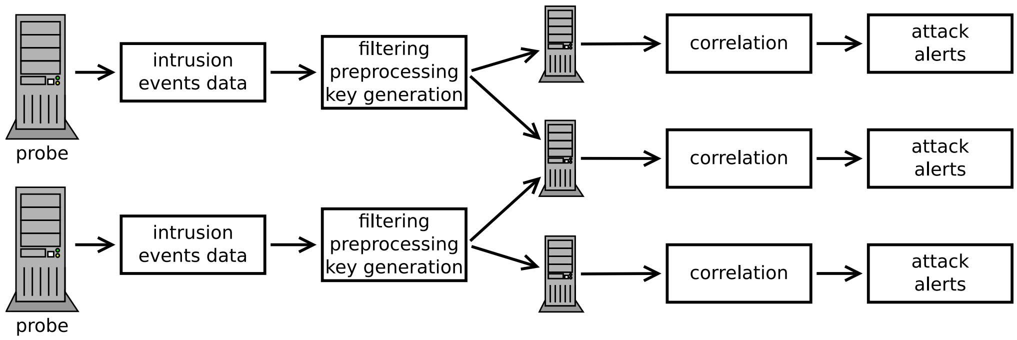

Figure 2. Distributed collection and distributed correlation of intrusion evidence from various probes. The Komondor system assigns keys to pieces of evidence so that data can be stored efficiently in a DHT. By using these keys, computational load of correlating can be distributed among several units.

Overlay	**Chord**	**Kademlia**
Routing algorithm	recursive	iterative
Node lookup	0	$\log_2 N$
First event stored	$\log_2 N$	$1 + \log_2 N$
n **events with the same key**	$n \cdot \log_2 N$	$n + \log_2 N$
Average number of messages per event	$(n \cdot \log_2 N)/n$	$(n + \log_2 N)/n$
Average number of messages with $n \to \infty$	$\log_2 N$	1

Table I
NUMBER OF MESSAGES IN STRUCTURED OVERLAYS FOR INTRUSION DETECTION

being an example for hacing rigid routing tables. Chord uses a *recursive routing mechanism,* which means that messages are forwarded by overlay nodes along the path from the source to the destination of the message. If Komondor would be built on Chord, the number of messages generated in the overlay would be in the order of $\log_2 N$ for each detected event, where N is the node count of the overlay.

Kademlia uses an iterative algorithm. To store a $\langle key; value\rangle$ pair, a Kademlia node first looks up the IP address of the destination node by successively querying nodes closer to the destination. After finding out its address, data is sent directly from the source and the destination. This also implies that the payload of the message is contained in every message for Chord, and only in the last message for Kademlia. For Kademlia, the node has to first look up the address of the destination, which also takes $\log_2 N$ messages. Having done that, it requires one more message $(+1)$ to send the payload as well. If multiple events are to be stored, which are detected by the same probe (this is a likely scenario for a node that is under attack), the *lookup procedure can be optimized away*, as the key and therefore the collector node is the same, too. For sending data of n events, the number of messages generated is only $n+\log_2 N$ for Kademlia and $n \cdot \log_2 N$ for Chord, which is worse at the factor of n for the latter one. The limit of messages per event drops to 1 for Kademlia in this common intrusion detection scenario.

IV. Results and Discussion

In this section, we present statistics of intrusion attempts detected using the implemented Komondor system. The statistics are evaluated to show which types of attacks this system can be used to detect.

The present Komondor implementation used the open-source *Snort intrusion detection system* [17] to detect intrusion events. However, it could collaborate with other intrusion detection solutions as well. The key selected for each event was the *IP address* of the attacker, as found in the Snort log file. It was also used for correlation. We selected common event types from the Snort database and also tagged events with a severity score. Intrusion alert was triggered when the sum of these scores reached a threshold level. This simple correlation method enabled us to determine the efficiency and reliability of the Komondor system for known attack types presented here. The number of probes in the system varied from 7 to 10, each with their own IP address but on the same subnetwork. Data presented here was collected in a three year interval.

A. Attack Intervals and Number of Events

Figure 3 shows invalid passwords detected for SSH login attempts on various hosts [3]. Every dot on the graph is an individual attack. The y axis shows the number of events or the number of invalid passwords detected. The duration of an attack is the time interval between the first and the last event detected, and is on the x axis. Several attackers were detected by multiple Komondor probes, because the SSH worm that was trying to gain access to the subnetwork tried to login all on-line hosts it found. The number of probes which detected an attack in question is shown by the color of the dots. (In the case of multiple probes detecting an attacker, the event number on axis x is an average per probe.)

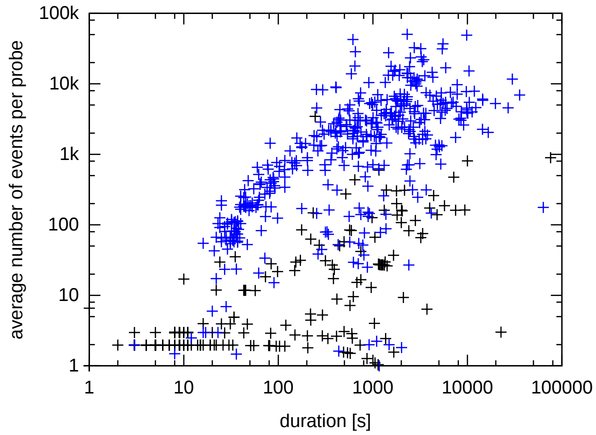

Figure 3. Number of invalid password events detected for various attacks (y axis) plotted by the duration of the attack (x axis). The color of the dots represent the number of probes a specific attacker was detected by.

Attacks, which were detected by one probe only (black dots) have much less events associated to them. The 1 100 attacks shown on the graph have as much as 450 of them stacked up in the $(1; 1)$ point. These evidently came from human interaction. Attacks detected by multiple probes usually suggest automatic worm programs using dictionary attacks against the detector hosts.

This experience suggests that distributed intrusion detection can benefit from the advantages of DHTs:

- Attackers could be detected by several probes at the same time. When multiple hosts are attacked, recognizing an attacker using any evidence from any probe of the Komondor network, several hosts could be protected using firewalls at the same time, which might promptly be attacked, too.
- Attack evidence came from multiple probes. One attack is likely to be associated to thousands or tens of thousands of events, which must be stored and processed in the overlay. This type of load can be dealt with the DHT fairly well, as it can select different collector nodes for each individual attack and therefore balance the load.
- When detecting an event, which generates the same key, the Kademlia DHT can significantly reduce network traffic, as the IP address of the collector nodes have to be looked up only once. When the IP address is obtained, the system works as if it were using a centralized approach with the same benefits as those.

B. Attack Types Detected by Komondor

Table II shows various attack types and the efficiency for the Komondor system regarding protection. The *protection* column shows the number of attacks for each type, for which the attack continued after it was blocked on the firewall, and the activity of the attacker was detected by another Komondor node of the same subnetwork. For these attacks, the collaborative intrusion detection can greatly enhance the protection of hosts.

Type of attack	Attacks	Protection	Ratio
phpMyAdmin scan	107	71	66%
MSSQL overflow	4355	15	0%
SSH connection lost	490	321	65%
SSH failed password	546	219	40%
SSH invalid user	51	47	92%
FTP failed login	46	2	4%

Table II
NUMBER OF ALL ATTACKS AND ATTACKS FOR WHICH PROTECTION COULD BE BUILT BY KOMONDOR, FOR EACH ATTACK TYPES.

Figure 4 shows event numbers and attack durations for different worms attacking SQL servers. The y axis has two scales for each graph. The scales of the left hand side show attack durations (red plot), and the right hand side scale shows the number of events (blue plot). Attacks are sorted by duration. Every value on the x axis is an attack for which the duration and the number of events is shown right under each other.

A worm, which scanned the Web servers for vulnerabilities via HTTP requests is shown on the right hand side subfigure. For any event detected, the IP address of the attacker can be recognized by the correlation units. The left hand side graph presents the properties of the Slammer worm, which penetrates outdated MSSQL servers. This worm does not issue more attempts in a short time interval to the same host, and selects IP addresses of victims randomly. For detecting this type of attacks, the PROMIS and CIDS systems could be used more effectively.

V. CONCLUSION

Attacks on the Internet mean constantly growing problem for network administrators. Sophisticated attacks have evidence spread across multiple hosts and subnetworks. To detect these attacks promptly and correctly, data must be aggregated and analyzed automatically. In this article, the novel Komondor intrusion detection system is presented, which enables current attack correlation methods to be upgraded to work in a distributed environment, thereby making them feasible for large-scale deployment. This is achieved by inserting a middle layer into the intrusion detection data path, which utilizes the Kademlia overlay.

The novelty of the method presented is attaching a key to the detected events, which key is then used to send the events for correlating to several correlation units that are organized as a DHT. This mechanism can be used to reduce network and computational load and increase reliability of the system, while still retaining the advantages of centralized approaches of intrusion detection. By mapping the detected

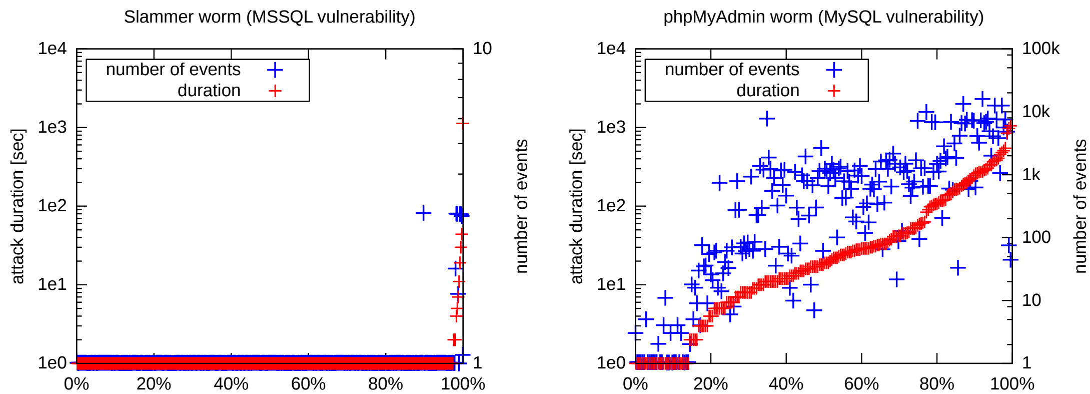

Figure 4. Attack intervals and number of events for different worm activities detected by the Komondor system. The left hand side shows a worm, which scanned our Web servers via HTTP in order to find a phpMyAdmin installation to gain access to MySQL databases. On the right hand side the activity of the infamous Slammer worm is shown, which penetrates MSSQL servers.

events to nodes in the system, all nodes are assigned the same level of responsibility as well. Our further research will focus on considering the different computational and network capacity of nodes to prevent those with slow connections or CPUs from being overloaded by intrusion detection data.

ACKNOWLEDGEMENT

The work reported in the paper has been developed in the framework of the project ”Talent care and cultivation in the scientific workshops of BME”. This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009.

REFERENCES

[1] D. Mutz, G. Vigna, and R. Kemmerer, “An Experience Developing an IDS Stimulator for the Black-Box Testing of Network Intrusion Detection Systems,” in *In Annual Computer Security Applications Conference, Las Vegas, NV*, 2003, pp. 374–383.

[2] H. Debar, “Intrusion Detection Systems-Introduction to Intrusion Detection and Analysis,” *Security and privacy in advanced networking technologies*, p. 161, 2004.

[3] C. Seifert, “Analyzing Malicious SSH Login Attempts,” http://www.symantec.com/connect/articles/analyzing-malicious-ssh-login-attempts, Nov. 2010, retrieved: March, 2012.

[4] C. Zhou, C. Leckie, and S. Karunasekera, “A Survey of Coordinated Attacks and Collaborative Intrusion Detection,” *Computers & Security*, vol. 29, no. 1, pp. 124–140, 2010.

[5] “Internet Storm Center,” http://www.dshield.org/, retrieved: March, 2012.

[6] R. Kemmerer, “NSTAT: A Model-based Real-time Network Intrusion Detection System,” *University of California-Santa Barbara Technical Report TRCS97*, vol. 18, 1997.

[7] A. Valdes and K. Skinner, “Probabilistic Alert Correlation,” *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pp. 54–68, October 2001.

[8] C. V. Zhou, S. Karunasekera, and C. Leckie, “A Peer-to-Peer Collaborative Intrusion Detection System,” in *Networks, 2005. 13th IEEE International Conference on*, vol. 1.

[9] F. Cuppens and R. Ortalo, “LAMBDA: A language to model a database for detection of attacks,” in *Recent advances in intrusion detection*. Springer, 2000, pp. 197–216.

[10] S. Templeton and K. Levitt, “A Requires/provides Model for Computer Attacks,” in *Proceedings of the 2000 workshop on New security paradigms*. ACM, 2001, pp. 31–38.

[11] V. Yegneswaran, P. Barford, and S. Jha, “Global Intrusion Detection in the DOMINO Overlay System,” in *Proceedings of NDSS*, vol. 2004, 2004.

[12] V. Vlachos and D. Spinellis, “A PRoactive Malware Identification System based on the Computer Hygiene Principles,” *Information Management and Computer Security*, vol. 15(4), pp. 295–312, 2007.

[13] R. Janakiraman, M. Waldvogel, and Q. Zhang, “Indra: A Peer-to-peer Approach to Network Intrusion Detection and Prevention,” in *Enabling Technologies: Infrastructure for Collaborative Enterprises. WET ICE 2003.* IEEE, 2003, pp. 226–231.

[14] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[15] S. Androutsellis-Theotokis and D. Spinellis, “A Survey of Peer-to-peer Content Distribution Technologies,” *ACM Computing Surveys (CSUR)*, vol. 36, no. 4, pp. 335–371, 2004.

[16] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the xor metric,” *Peer-to-Peer Systems*, pp. 53–65, 2002.

[17] “Snort – Open-source Intrusion Detection System,” http://www.snort.org/, retrieved: March, 2012.

 ISBN: 978-1-61208-201-1