available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A survey of coordinated attacks and collaborative intrusion detection

*Chenfeng Vincent Zhou**, *Christopher Leckie*, *Shanika Karunasekera*

Department of Computer Science and Software Engineering, The University of Melbourne, 111 Barry Street, Carlton, Victoria 3053, Australia

ARTICLE INFO

Article history:
Received 17 April 2009
Received in revised form
4 June 2009
Accepted 29 June 2009

Keywords:
Network security
Threat
Intrusion detection
IDS systems and platforms
Assessment

ABSTRACT

Coordinated attacks, such as large-scale stealthy scans, worm outbreaks and distributed denial-of-service (DDoS) attacks, occur in multiple networks simultaneously. Such attacks are extremely difficult to detect using isolated intrusion detection systems (IDSs) that monitor only a limited portion of the Internet. In this paper, we summarize the current research directions in detecting such attacks using collaborative intrusion detection systems (CIDSs). In particular, we highlight two main challenges in CIDS research: CIDS architectures and alert correlation algorithms. We review the current CIDS approaches in terms of these two challenges. We conclude by highlighting opportunities for an integrated solution to large-scale collaborative intrusion detection.

1. Introduction

The openness and scalability of the Internet has made it a flexible platform for a new generation of on-line services, such as electronic commerce, entertainment and social networking. The popularity of these services has resulted in a huge volume of financial transactions and other types of sensitive information being accessed via the Internet (Kruegel, 2005). However, the importance and value of this information and the related on-line services have made the Internet a target for a wide variety of attacks, which threaten the security of the Internet. For example, the number of reported Internet security incidents has jumped from 6 in 1988 to 137,529 in 2003, and the total number of cataloged vulnerabilities has increased over 42 times from 171 in 1995 to 7236 in 2007 (CERT, 2006).

In recent years, attackers have shown increasing sophistication in their ability to launch attacks that target or utilize a large number of hosts that are spread over a wide geographical area or multiple administrative domains (CERT, 2003b). For example, attackers can scan large numbers of hosts simultaneously to search for software vulnerabilities (*i.e.*, stealthy scans); they can use self-replicating computer programs to spread their malicious code to many thousands of vulnerable systems within a short time period (*i.e.*, worms); and they can use thousands of compromised hosts from different network domains to overload a targeted link or system to disrupt its service (*i.e.*, distributed denial-of-service (DDoS)). We refer to these types of attacks as *large-scale coordinated attacks*.

These coordinated attacks pose a significant threat to the security of the Internet. For example, in 2003 the SQL-Slammer worm infected 75,000 hosts in 10 min, which caused significant disruption to financial, transportation, and government institutions (Moore et al., 2003). On 19th January 2007, the Storm worm infected thousands of computers in Europe and the United States (Symantec Threat Advisory Center, 2007). From

* *Corresponding author.*
E-mail addresses: cvzhou@csse.unimelb.edu.au (C.V. Zhou), caleckie@csse.unimelb.edu.au (C. Leckie), shanika@csse.unimelb.edu.au (S. Karunasekera).
0167-4048/$ – see front matter
doi:10.1016/j.cose.2009.06.008

September 2007 to March 2008, there were an average of 1300 distributed denial-of-service attacks occurring each day according to Arbor Networks (McPherson, 2008).

These coordinated attacks are extremely difficult to detect since the evidence of the attacks is spread across multiple network administrative domains. Consider the illustrative example of a coordinated attack in Fig. 1, which depicts a stealthy scan. In this figure, we show six network domains that are each connected to the Internet via a local intrusion detection system (IDS). An *attack host* sends probe packets to test for potentially vulnerable *target systems* on each network. If a large number of probe packets are seen at a single target network over a given time interval, then the IDS at that network has a high degree of confidence that a scan is occurring, and can raise an alert. However, if the scanner randomizes the probes between multiple target networks, then in a given time interval each IDS will see fewer probes, and will be less likely to be detected by the local IDS in each network. Similar scenarios occur during the early stages of a worm outbreak, when an attack host starts to infect target systems with malicious code, or during a DDoS attack, when a master node sends commands to compromised hosts to launch denial-of-service traffic against an on-line service.

In order to detect these types of large-scale coordinated attacks, we need the ability to combine the evidence of suspicious network activity from multiple, geographically distributed networks. Rather than each IDS working in isolation, we can form a *collaborative intrusion detection system* (CIDS), which analyzes evidence from multiple networks simultaneously. By combining evidence from multiple networks, we can detect such coordinated attacks at an earlier stage, before they have had a significant impact on the Internet. Although coordinated attacks may be easier to detect at a later stage when the volume of attack traffic is large, the utility of intrusion detection would be diminished because by that stage the damage has been done.

In this paper, we first examine different types of coordinated attacks, such as large-scale stealthy scans, worm outbreaks and distributed denial-of-service (DDoS) attacks. We then review the state-of-the-art in two main challenges of CIDS research: *system architecture* and *alert correlation*. This paper does not explore in detail machine learning based data mining methods for intrusion detection, DoS defense and network wide anomaly detection, which have been previously surveyed in (Brugger, 2004; Lee and Stolfo, 2000; Stolfo et al., 2001; Mirkovic and Reiher, 2004; Peng et al., 2006; Lakhina et al., 2004, 2005).

The remainder of this paper is organized as follows. In Section 2, we first review the types of coordinated attacks that have been observed on the Internet, in order to identify the characteristic features of these attacks. Then we describe the existing research on detecting these types of attacks using collaborative intrusion detection systems (CIDSs) in Section 3. In particular, we highlight two main challenges in CIDS research: CIDS system architectures, and alert correlation algorithms. In Section 4, we review the current CIDS approaches in terms of these two challenges. We highlight the opportunities for an integrated CIDS in Section 5, and conclude in Section 6.

2. Coordinated attacks

The motivation driving hackers on the Internet has shifted from the pursuit of notoriety to the pursuit of profit. Rather than boosting their prestige by defacing websites, hackers now use spam – indiscriminately sending out unsolicited bulk messages, phishing – a form of web-based identity theft, or DDoS extortion for monetary gain (Chiueh, 2007; Franklin et al., 2007). New vulnerabilities (known as zero-day exploits) are routinely bought and sold by underground organizations (Miller, 2007). Large-scale coordinated attacks, such as

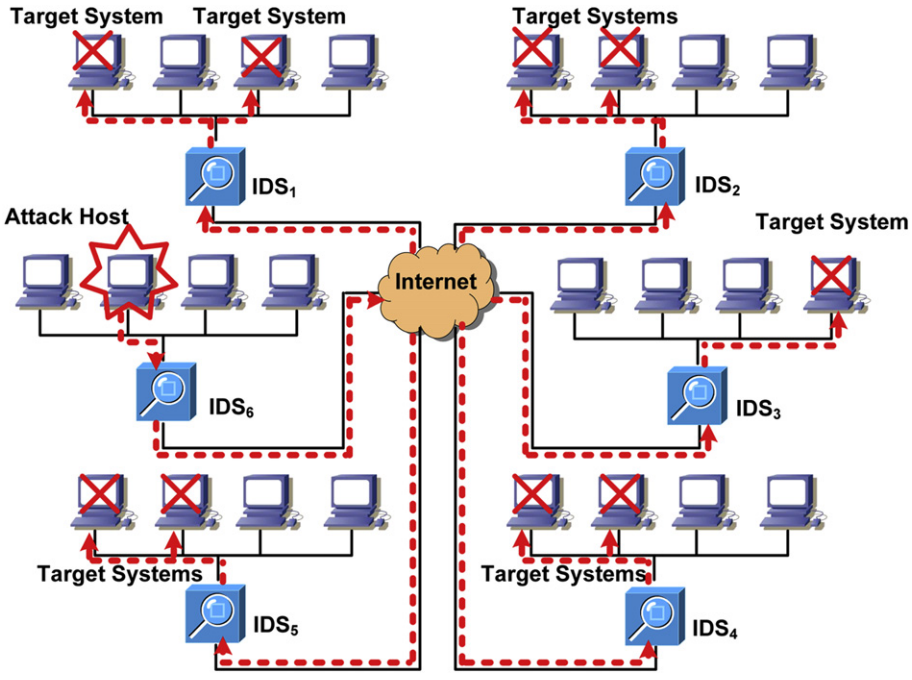

Fig. 1 – A common stage of coordinated attacks.

stealthy scans, worms and DDoS attacks, are powerful tools to assist attackers to achieve monetary gain (CERT, 2003b). Attackers can nowadays use automation to target all vulnerable services at once, rather than manually target high value services (Savage, 2005). These large-scale attacks can occur in multiple network domains simultaneously, which makes prompt detection an extremely difficult task.

A typical attacking process using coordinated attack techniques can be described as follows.

1. The attacker uses large-scale stealthy scans (which can evade local IDSs) to hunt for known software vulnerabilities or to generate a list of vulnerable addresses that can be used for spreading a worm later (*e.g.*, using a flash worm Staniford et al., 2004).
2. Using a known vulnerability or a target list acquired by the stealthy scans or bought from the vulnerability market (Miller, 2007), the attacker writes a worm script to compromise hundreds or thousands of hosts simultaneously by exploiting this vulnerability. The compromised hosts (also known as ''zombies'' or ''bots'' – a compromised computer system controlled by an attack) can then be used to launch a DDoS attack or can be sold on the underground market.
3. After recruiting tens of thousands of compromised hosts through worm spread or from underground markets, the attacker can use large-scale DDoS attacks to blackmail service providers for monetary gain (Franklin et al., 2007).

In this section, we review three main coordinated attack methods: large-scale stealthy scans, worm outbreaks and DDoS attacks. We describe each attack type in terms of the attack's purpose, the vulnerabilities exploited, the attack frequency and the attack topology, in order to understand the motivation behind these attacks, and to identify the common characteristics of these attacks upon which a detection approach can be built. Note that we do not attempt to provide a comprehensive survey of these attacks, which can be found in (Staniford, 2002; Weaver et al., 2003; Goldi and Hiestand, 2005; Peng et al., 2006; Mirkovic and Reiher, 2004; Igure and Williams, 2008).

2.1. *Large-scale stealthy scans*

The primary intention of an attacker who performs a large-scale stealthy scan is to gather information about the reachability and the status of particular ports and IP addresses of interest (Staniford, 2002). In order to evade being detected by a local IDS, various techniques are used in stealthy scans, such as randomizing scan order to evade detection algorithms that test for sequential scans of a range of IP addresses, or slowing down the scan frequency to defeat scan detectors that count the number of related probes within a given detection window.

There are two main types of large-scale stealthy scans: horizontal scans and block scans (Staniford, 2002). Horizontal scans are used to scan for a certain port of interest on all IP addresses in some range, in order to locate a vulnerability in a particular service that can be exploited. For example, an attacker may scan TCP port 139 in sequential IP addresses to exploit a NetBIOS vulnerability. Block scans are used to scan a set of services on a range of hosts, in order to identify numerous services on multiple hosts. For example, for a list of interested IP addresses, the attacker may scan all the ports on each address to identify which services are currently active, *i.e.*, which vulnerabilities can be exploited later.

The topology of these attacks is from one attack host to many target hosts. By randomizing a scan across multiple networks simultaneously, there is less likelihood that any single local IDS can observe a sufficient number of scan accesses within its detection window. Thus without collaboration between these IDSs, it is extremely difficult to detect these attacks.

2.2. *Worm outbreaks*

A worm is a self-replicating computer program that can send copies of itself to other computers in the network without user intervention. An attack that aims to spread a worm on the Internet has two main purposes: (1) to cause a traffic overload in local area networks and congestion on Internet links, which disrupts affected hosts and leads to financial losses; and (2) to recruit compromised hosts for future use. Nowadays, since the motivation of Internet attacks has shifted from notoriety to financial gain, most attackers do not want to draw attention to their actions by causing unnecessary network disruption (Chiueh, 2007). However, as new worms are discovered from time to time, they remain a threat to the security of the Internet. For example, the Storm worm began infecting thousands of computers with the Microsoft Windows operating system in Europe and the United States on 19th January 2007, using an e-mail message with a subject line about a recent weather disaster (Symantec Threat Advisory Center, 2007). The latest version of the Storm worm was released onto the Internet on 1st April 2008 (Nichols, 2008). Nevertheless, the basic operation of worm attacks remains the same. Worm spread consists of two phases: an infection phase and a propagation phase. The infection phase is used to scan for vulnerable hosts and then exploits the vulnerability to prepare for the second phase. The propagation phase is then used to transmit the worm code to the target hosts (Goldi and Hiestand, 2005). The attack topology of all these worms comprises one attack host to many target hosts during the initial worm spread. In this section, we survey three well-known examples of worms: SQL-Slammer, Code Red 2 and W32/Sasser.

SQL-Slammer began to infect hosts slightly before 05:30 UTC on 25 January 2003, by exploiting a buffer-overflow vulnerability in computers running Microsoft SQL Server or Microsoft SQL Server Desktop Engine (MSDE) 2000 (Moore et al., 2003; CERT, 2003a). As it began spreading through the Internet, the worm infected over 90 percent of vulnerable hosts within 10 min. The spreading process was as follows. The worm first crafts UDP packets of 376-bytes and sends them to randomly chosen IP addresses on port 1434. If the packet is sent to a vulnerable machine, this victim machine will become infected and will also begin to propagate. The maximum rate of propagation achieved was more than 55 million scans per second (Moore et al., 2003).

Code Red 2 appeared and started to spread on 19 July 2001, by exploiting a buffer-overflow vulnerability in Microsoft IIS Server Indexing Service DLL (CERT, 2001). It infected over

359,000 machines within 14 h (Moore et al., 2002). The spreading process was as follows. The worm first generates 99 infection threads in parallel, and each thread attempts to connect to TCP port 80 on a randomly chosen host. Upon a successful connection to port 80, the attacking host sends a crafted HTTP GET request to the victim, attempting to exploit a buffer-overflow vulnerability in the Indexing Service. On successful completion of the process, the worm begins executing on the victim host and starts to propagate itself.

The W32/Sasser worm was first documented on 1 May 2004, when it spread itself on the Internet by exploiting a buffer-overflow vulnerability in the Windows Local Security Authority Service Server (LSASS) (CERT, 2004). This caused significant performance degradation on the infected systems. According to Brenner (2004), the W32/Sasser worm had infected around one million unpatched hosts by 10 May 2004. The spreading process was as follows. The worm first scans random IP addresses on TCP port 445 to identify vulnerable systems. Upon establishing the TCP connection successfully, the worm then exploits the LSASS vulnerability on the located vulnerable system by creating a remote shell on TCP port 9996. Then an FTP server on TCP port 5554 is started using the shell to download a copy of the worm from the attacking system. The worm then begins executing on the victim host and starts to propagate itself.

2.3. *Distributed denial-of-service attacks*

A DDoS attack is a powerful tool used by attackers to disrupt the on-line service of a victim's web service, which is motivated mainly by monetary gain either by extorting the victim or by a payment from an opponent of the victim (Chiueh, 2007). In comparison to a traditional DoS attack that uses a single attack source, DDoS attacks use multiple attack sources, which amplifies the attack power and makes defense more complicated. On 7 February 2000, a DDoS attack successfully disrupted Yahoo, Buy.com, eBay, Amozon.com, CNN.com and several other websites, which resulted in substantial damage (Garber, 2000). According to data gathered by Arbor Networks (McPherson, 2008) from September 2007 to March 2008 via its Atlas program, there are an average of 1300 distributed denial-of-service attacks occurring each day. A DDoS attack can be split into two stages: recruiting and attacking. In the first stage, an attacker hunts for vulnerable hosts on the Internet and installs attack tools on them. The attack topology in this stage is often one attack source to many target systems, which is similar to the infection phase of worm spread. In the second stage, the attacker sends attack commands to these compromised systems to launch an attack against a third party (Dietrich et al., 2000). The attack topology in this stage is often many attack sources to one target system. We survey two well-known DDoS attacks, distributed SYN floods and distributed reflector attacks as follows, focusing on the attack stage.

The distributed SYN flood exploits a vulnerability of the TCP three-way handshake, *i.e.*, a server needs to allocate a large data structure for any incoming SYN packet (CERT, 1996). During the attack, the compromised systems (''zombies'') are instructed to send SYN packets with an invalid source IP address, to create an instance of a half-open connection data structure on the target server. Eventually the memory stack on the target system is filled up and no new requests can be processed. In order to flood the memory stack on the target system, a huge number of attack requests are sent in a short period of time.

The distributed reflector attack uses innocent third party systems (routers or web servers) to relay attack traffic to the target in order to obscure the sources of attack traffic and to amplify the volume of attack traffic (Paxson, 2001). In this attack, the attacker not only controls an army of compromised hosts (''zombies''), but also maintains a list of *reflectors* (*i.e.*, any IP host that will return a packet if sent a packet, such as all web servers, DNS servers, and routers). During the attack, the attacker instructs its zombies to send the attack traffic with the target's IP address as the source IP address (*i.e.*, spoofed traffic) to the reflectors, rather than ordering the zombies to send attack traffic to the target directly. Then the reflectors complete the attack by sending the reply traffic to the victim. Note that if the zombies send a spoofed TCP SYN packet, the reflector will send a SYN-ACK packet to the target, which may be retransmitted several times by the reflector if there is no reply. As a consequence, the volume of attack traffic will be *amplified* by the reflectors. In this case, each reflector will see the same spoofed source IP address in the incoming SYN packets (Paxson, 2001).

2.4. *Summary*

The coordinated attacks described above are summarized in Table 1. As shown in this table, large-scale stealthy scans and

Table 1 – Summary of coordinated attacks.

Attack type		Attack purpose	Vulnerabilities exploited	Traffic rate	Attack topology
Large-scale stealthy scans		Hunting for vulnerable hosts	Naive detection algorithm	Low	One to many
Worm outbreak	SQL-Slammer	Recruiting zombies	MS SQL Server or MSDE 2000	High	One to many
	Code Red 2	Recruiting zombies	MS IIS Web Server	High	One to many
	W32/Sasser	Recruiting zombies	LSASS	High	One to many
DDoS	Distributed SYN flood	Monetary gain	TCP 3-way hand-shake	High	One to many & many to one
	Distributed reflector attacks	Monetary gain	N/A	High	One to many & many to one

worm outbreaks use the same topology during an attack. In particular, there is one source (attacking host) that is responsible for numerous scans during the large-scale stealthy scans. Similarly, there is one source (infected host) that begins to connect to numerous hosts in order to spread itself during the worm outbreak. In contrast, the attack topology of a DDoS attack is many to one, namely, all the attack traffic is forwarded to one destination (the target system), although in a distributed reflector DDoS attack, part of the attack topology may appear as one to many (*i.e.*, one spoofed source to many reflectors). Therefore, in order to detect the source address of a stealthy scan or worm outbreak, we need the ability to correlate suspicious source addresses from incoming traffic across multiple network domains. Similarly, to detect and filter DDoS traffic we either need to correlate traffic at its source based on a common destination address, or correlate traffic at the reflectors based on a common (spoofed) source address. Moreover, given that the attack rate is high (except for stealthy scans) as shown in Table 1, this correlation of attack evidence must be done in a timely manner. In the next section, we summarize the challenges for collaborative intrusion detection in the context of these coordinated attacks.

3. Collaborative intrusion detection systems

In order to detect the coordinated attacks described in the previous section, CIDSs have been proposed to correlate suspicious evidence between different IDSs to improve the efficiency of intrusion detection. CIDSs have the potential to detect intrusions that occur across the whole Internet simultaneously by correlating attack signatures among different subnetworks of the Internet. They also have the potential to reduce computational costs by sharing intrusion detection resources between networks. The main objective of CIDSs is to reduce the number of false alarms and irrelevant alerts that would be generated by individual IDSs acting in isolation, and produce a high level overview of the security state of the whole network by correlating the alerts from individual IDSs. A CIDS consists of two main functional units:

1. A *detection unit*, which consists of multiple detection sensors, where each sensor monitors its own subnetwork or hosts separately and then generates low-level intrusion alerts.
2. A *correlation unit*, which transforms the low-level intrusion alerts into a high level intrusion report of confirmed attacks.

3.1. Challenges of collaborative intrusion detection

CIDSs have the potential to resolve the problems of isolated IDSs, since they are able to identify network wide attacks and reduce false alarms by combining evidence of attacks from multiple networks. However, CIDSs introduce new challenges as follows.

- *System architecture*: a CIDS is essentially a distributed intrusion detection system. Therefore, the architecture determines how the alerts from individual detection sensors are being shared and processed. Where to place the detection unit and correlation unit will affect the scalability and performance of the CIDS.
- *Alert correlation*: the main goal of a CIDS is to detect network wide attacks and reduce irrelevant alarms, which is achieved by alert correlation (*i.e.*, the data correlation unit). How the alerts from individual sensors are correlated determines the detection accuracy of a CIDS.
- *Data privacy*: data privacy is an important issue in practice if information is being shared between organizations. If appropriate privacy measures are not provided by a CIDS, then the individual participants are unlikely to share their alerts in the first place.
- *Security and trust*: like other distributed systems, security and trust is an important aspect for any CIDS. Since the overall detection accuracy of the CIDS depends on the correctness of the alert information provided by each participating IDS, it is important to verify the trustworthiness of the alerts.

4. Existing collaborative intrusion detection classifications

There are many ways to classify CIDSs. We divide current collaborative intrusion detection proposals according to the two challenges for CIDSs listed above, *i.e.*, *system architecture* and *alert correlation*. We use examples of prominent systems to explain the strengths and weaknesses of existing CIDSs in each category, rather than providing an exhaustive survey. Note that while other issues, such as privacy and trust, are relevant to CIDSs, they are not the main focus of this paper, and will only be briefly discussed.

4.1. System architecture

Various schemes have been proposed to enable the effective aggregation and correlation of information from individual IDSs in a CIDS. These schemes can be classified into three groups, *centralized approaches* – where all the information collected from each IDS is reported to a single location for analysis; *hierarchical approaches* – where local information is pre-processed, then selected information is reported to the next layer in a hierarchy for sharing and further analysis; and *fully distributed approaches* – where the information from each IDS is shared and processed in a completely distributed fashion without a centralized coordinator.

4.1.1. Centralized approaches

Fig. 2 depicts an overview of the centralized correlation approach for a CIDS with eight participating IDSs. In this approach, each IDS plays a role as a *detection unit* in the CIDS, where it produces alerts locally. Then the alerts are reported to a central server that works as a *correlation unit* for analysis. We present three prominent examples of centralized approaches: DIDS, DShield and NSTAT.

DIDS (Snapp et al., 1991) combines multiple IDSs running on individual systems to detect network-wide intrusions. There is a single host monitor per host and a single LAN monitor for each broadcast LAN segment in the monitored

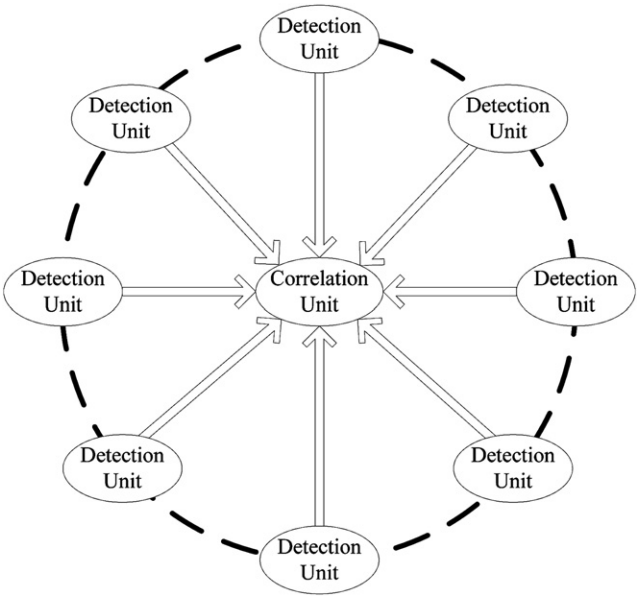

Fig. 2 – Centralized CIDS architecture.

network in the DIDS. Each local IDS collects any suspicious information locally and converts it to a homogeneous format, then reports the generated alerts to a centralized rule-based expert system for further evaluation. Alerts are generated if any rule has been satisfied after correlation. However, clever attackers can evade DIDS by reducing the attack traffic for a given network.

DShield (Internet Storm Center) is a cooperative network security community, which collects firewall and IDS logs world wide. Users of DShield can submit their firewall/IDS logs to the DShield database via either the client software or the Web interface provided by DShield. Then the collected intrusion logs are centrally processed and analyzed in the DShield database. The analysis results are used to generate security reports such as the top reported ports or top sources of attacks, and predict the emerging security trends on the Internet.

NSTAT (Kemmerer, 1998) is another example of distributed collection and centralized correlation. NSTAT uses a client server architecture to detect network-wide intrusions. The client side is responsible for reading and filtering the audit trail on local hosts and then sending the collected data to a central server. The server merges the data from various sources into a single chronological audit trail, then performs the analysis using a state transition based mechanism to detect intrusions (where penetrations are predefined as a series of state transitions described in terms of signature actions and state assertions). There are also some early prototypes such as (Heberlein et al., 1992; Hochberg et al., 1993; Mounji et al., 1995; Huang et al., 1999) that collect their data in a distributed fashion and analyze them centrally.

These centralized approaches are usually suitable for small enterprise scale cooperation, but not for independent IDSs on the Internet. There are two primary shortcomings of this centralized approach. First, the central unit becomes a single point of failure or vulnerability. The correlation process of the CIDS will be completely deactivated if the central server fails. Second, although individual alerts are usually reduced before being sent to the central correlation unit, the processing capacity of the central node will limit the volume of data it can handle in a given amount of time. When increasing numbers of alerts are forwarded to the central unit when more IDSs join the CIDS or when the monitored networks are undergoing an attack, the centralized approaches often have slow response times or may suffer from data loss.

4.1.2. *Hierarchical approaches*

In order to address the scalability problem of the centralized approach, several hierarchical designs have been introduced. In this approach, the entire CIDS is partitioned into multiple small communication groups based on one of the following features: (1) *geography*, (2) *administrative control*, (3) *collection of similar software platforms*, and (4) *anticipated types of intrusions*. Each communication group is a subset of the hierarchy. There is an analysis node in each group that is responsible for correlating all the data collected in this group. This analysis node is the parent node of the group, and its processed data will be sent upward to a node at a higher level in the hierarchy for further analysis. Fig. 3 depicts an overview of the hierarchical correlation approach for a CIDS. The system is divided into three communication groups as described above. Normally, IDSs in the lowest level work as *detection units*, and their alerts are passed upward for correlation. IDSs in the higher level are equipped with both a *detection unit* and *correlation unit*. They correlate alerts from both their own level and their children nodes. Then the correlated alerts are passed upward to a higher level for further analysis.

GrIDS (Staniford-Chen et al., 1996) is a graph-based hierarchical CIDS, which aims to detect network-wide intrusions that involve connections between many nodes. The network monitored by GrIDS is divided into small communication groups as described above, to build a hierarchical tree. Each small group has an analysis node (parent node) that is responsible for collecting data about activity on hosts (its children nodes) and network traffic between them. All the data is then aggregated into activity graphs. These graphs are able to reveal the causal structure of network activity. The nodes in the graph represent hosts or the small communication groups, and the edges represent network traffic between them. Nodes and

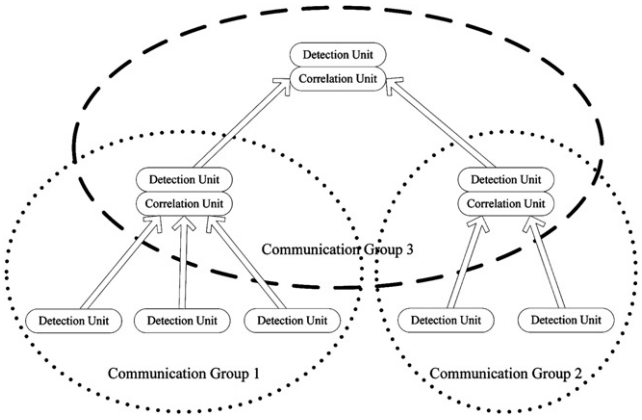

Fig. 3 – Hierarchical CIDS architecture.

edges are annotated with attributes (*e.g.*, the count of the activity). Each small group builds its own graphs and then passes upward to its parent a *reduced graph* where the whole group might be represented by a vertex. The graphs hence become coarser as they are propagated upward. Detection rules are used to determine if a graph is suspicious (*e.g.*, a certain attribute of the activity exceeds a user specified threshold).

The EMERALD project (Porras and Neumann, 1997) introduces a hierarchical framework to detect intrusions within a large enterprise network. The protected network is divided into three abstract layers: *service layer*, *domain-wide layer*, and *enterprise-wide layer*. Each layer includes independently tunable monitors that combine signature analysis with statistical profiling. The service layer, which is the lowest layer in the hierarchy, is responsible for monitoring the individual components and network services within a single domain. The monitors in the domain-wide layer monitor and analyze misuse across multiple services and components. The highest layer in the hierarchy tree of EMERALD is the enterprise-wide layer, which analyzes misuse across multiple domains in the entire system. A subscription-based communication mechanism is used to coordinate the information dissemination. For example, each monitor is able to subscribe to information from other monitors at the same or lower level in the hierarchy to improve the detection capacity. This asynchronous communication model enables EMERALD to efficiently communicate the information where it is needed. However, without a structured overlay, ensuring reliable message delivery can create large communication overhead, especially in a large-scale network. Furthermore, the hierarchical architecture limits the types of intrusions that can be detected at the highest levels in EMERALD, although it allows the system to be deployed across large enterprise-scale networks.

Li et al. (2007) proposed a hierarchical CIDS based on dependency. Participating hosts are clustered into cooperating regions based on network knowledge (such as proximity, local host properties, policy constraints and boundaries). A hidden Markov model (Rabiner, 1989) is used to aggregate the alerts collected from local hosts within the region. Then sequential hypothesis testing (Jung et al., 2004) is applied globally to correlate findings across regions. DSOC (Distributed Security Operation Center) (Abdoul Karim Ganame et al., 2008) is a hierarchical (three layer in particular) architecture for collaborative intrusion detection. The suspicious information is collected locally by the first layer devices: Data Collection Boxes (e.g., local firewall/IDS) and Remote Collectors (i.e., data collected from critical sensors). The second layer devices, Local Analyzers, are responsible for analyzing this suspicious information and generate alerts accordingly. Then the generated alerts are passed to the third layer device, the Global Analyzer (a dedicated Local Analyzer), for correlation and aggregation. Servin and Kudenko (2008) proposed a hierarchical CIDS based on reinforcement learning. This CIDS also has three layers, sensor agents for different network regions sitting at the first layer collect local abnormal information. The collected information is passed to the second layer – regional IDS agents which use reinforcement learning to analyze what information should be passed up to the third layer – a reinforcement learning based IDS. The third layer IDS can therefore generate alarms accordingly. AAFID (Balasubramaniyan et al., 1998) and NetSTAT (Vigna, 1999) are other examples of hierarchical CIDSs.

The hierarchical architectures scale better than the centralized approaches. However, the nodes of the higher levels in the hierarchy still limit the scalability of the CIDS, and their failure can stop the function of their whole subtree. Furthermore, the nodes in the highest level often have limited detection coverage due to the higher level of abstraction of the input data.

4.1.3. *Fully distributed approaches*

Several fully distributed CIDSs have been proposed to address the above problems of hierarchical approaches. Fig. 4 depicts an overview of the fully distributed architecture of a CIDS. In a fully distributed CIDS, each participant IDS has two function units: a detection unit that is responsible for collecting data locally; and a correlation unit that is a part of the distributed correlation scheme. The participant IDSs communicate with each other using some form of data distribution protocol, such as peer-to-peer (P2P), gossiping, multicast or publish/subscribe protocol.

Locasto et al. (2005) proposed a fully distributed CIDS based on a P2P architecture. Each participant uses an IDS to monitor its subnetwork or hosts. A tool called *Worminator* is run on the participating hosts at specified intervals to parse the alert output of the local IDS into the form of a *watchlist* (a list of suspicious IP addresses) and then to encode the *watchlists* in Bloom filters (Bloom, 1970). There are three benefits to using Bloom filters: compactness (reduction in data size), resiliency (no false negatives) and security (one-way data encoding). Next, the encoded *watchlists* are distributed over a decentralized P2P-style network among the participants. *Whirlpool*, which is a correlation group scheduling tool, is introduced to periodically reshuffle correlation groups, to approximate the optimal correlation schedule, *i.e.*, maximizing the bandwidth savings and minimizing the detection delay.

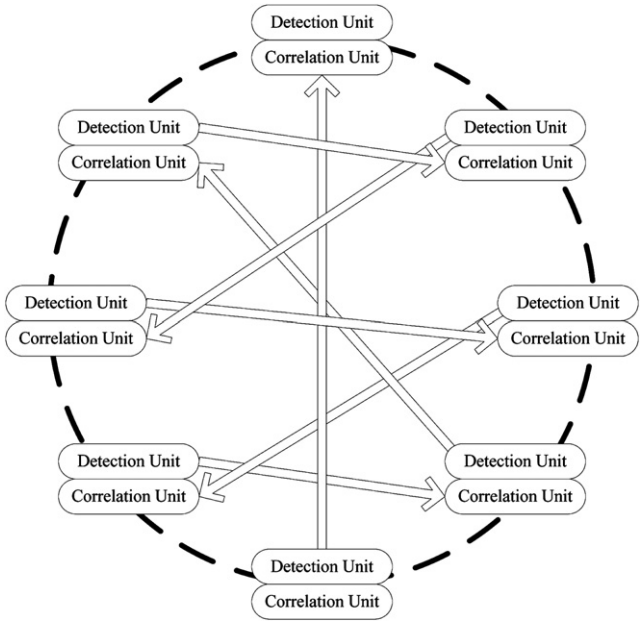

Fig. 4 – Fully distributed CIDS architecture.

The DOMINO project (Yegneswaran et al., 2004; Barford and Jha, 2004) is a distributed CIDS that aims to monitor Internet-scale outbreaks. There are three types of participants in the DOMINO network: *axis overlay* (the central component of DOMINO), *satellite communities* (components that provide a wide diversity of alert data) and *terrestrial contributors* (which are a means for expanding coverage by including external intrusion data sets). The nodes in the axis overlay are connected using a P2P protocol, and they participate in a periodic exchange of intrusion information. There are two views of current intrusion activity maintained in each axis node: (1) the local view, which is created from raw alerts from the monitored subnetwork and aggregated alerts from its satellite nodes; and (2) the global view, which is created by alert summaries periodically received from other axis nodes in the DOMINO system. An open issue in the DOMINO system is how to balance the trade-off between the detection rate versus the false alarm rate. Furthermore, the efficiency of the distributed architecture has not been evaluated in a large-scale deployment.

Dash et al. (2006) proposed a collaborative system of host-based IDSs, which use distributed probabilistic inference to detect slow network intrusions. There are three major components in the proposed system: (1) *local detectors* (LDs) are a local version of the IDS for detecting intrusions by analyzing the local state or local traffic patterns; (2) *global detectors* (GDs) are placed on the end-hosts but are responsible for generating global views of attacks by analyzing the information on the status of various LDs using probabilistic detector models; (3) *information sharing system* (ISS) uses a gossip protocol to communicate state between the LDs and GDs. The detection process is described as follows. Each LD uses a binary classifier to analyze both outgoing traffic from the host and incoming packets to the host. Alerts are raised if a configured threshold is crossed. The security state of a LD is beaconed to a random set of GDs at regularly spaced epochs by using the ISS. The GDs then generate a global view of the current security status of the monitored system by analyzing collected LD state information using a probabilistic detector model.

Garcia et al. (2004) proposed a distributed CIDS based on a multicast infrastructure to prevent coordinated attacks against a third party. The collaborative architecture consists of a set of cooperative entities, which work as independent *prevention cells* (local IDSs). These *prevention cells* detect attack steps locally, then exchange alert messages via a publish/subscribe system in order to work out appropriate countermeasures to prevent being an unwitting participant of a coordinated attack. MADIDF (Mobile Agents based Distributed Intrusion Detection Framework) (Dayong Ye et al., 2008) is a fully distributed CIDS based on mobile agents. Each participant host of MADIDF consists of four agents: a monitor agent, analysis agent, executive agent and manager agent. The framework also has two mobile agents: a retrieval agent and a result agent, which can travel among hosts. When the framework is under multi-host attack, the analysis agent detects the suspicious activity based on the information collected by the monitor agent, and sends a request to the manager agent for further investigation. Upon receiving the request, the manage agent dispatches a retrieval agent to gather relevant information from other hosts. Those hosts that are visited by the retrieval agent send a result agent back to the initial manager agent. The manager agent can therefore make a final decision based on the information received. Indra (Janakiraman and Zhang, 2003) is a P2P based CIDS which uses a neighborhood watch mechanism between trusted peers to share suspicious activity. CSM (White et al., 1996) is an early prototype of a distributed CIDS based on a P2P approach.

4.1.4. *Limitations of fully distributed approaches*

The fully distributed architecture addresses the scalability challenge of a CIDS. However, there are several open issues as follows.

- *Detection accuracy* – In a centralized CIDS, since all the alert information is available in the central server, an accurate detection decision can then be made based on all the information. However, depending on how information is disseminated in a fully distributed CIDS, not all alert information may be accessible at the location where the detection decision is made for a given attack. Therefore, it may be difficult to guarantee the same detection accuracy as a centralized CIDS, and we need to consider how to balance the trade-off between the detection rate and false alarm rate in the CIDS. Maximizing the detection rate and minimizing the false alarm rate are common goals of IDS research. However, most fully distributed approaches focus on the architectural aspect of the CIDS while ignoring detection accuracy.
- *Scalability* – Alert correlation is a key factor for the *effectiveness* of a CIDS. However, most fully distributed approaches use a single feature (such as the source IP address) to represent the alert information (*i.e.*, single-dimensional correlation), which is too restrictive to catch the important characteristics of large-scale attacks. Thus, we need to consider how to improve the scalability of more sophisticated alert correlation algorithms.
- *Load balancing* – In a fully decentralized CIDS, it is difficult to achieve load balancing across the CIDS, in such a way that no individual participating IDS becomes overloaded. Most distributed CIDSs in the literature use a source address based correlation scheme, which may create a load ''hot-spot''. For example, if a single source scans many subnetworks, this can create a flood of evidence to the responsible participating IDS. A load ''hot-spot'' in the CIDS may cause delays in correlation or even information loss. Moreover, an attacker can exploit this vulnerability by launching massive scans to multiple networks from a single source in order to overload the responsible participating IDS, and ultimately disrupt the CIDS.

Furthermore, to date, there has been little work on large-scale experiment to evaluate the performance of the existing distributed CIDS architectures, and in some cases, no form of evaluation has been conducted (*e.g.*, Garcia et al., 2004 and Janakiraman and Zhang, 2003). In general, the evaluation schemes for existing distributed CIDS architectures are as follows.

- Small scale deployment: *e.g.*, Locasto et al. (2004) evaluated their system by conducting experiments on four systems deployed at different sites in Northeast USA; Dash et al. (2006) tested their system on five weeks of traffic data for 37 hosts within their enterprise network.

- Simulation: *e.g.*, the reaction time in the DOMINO system (Yegneswaran et al., 2004; Barford and Jha, 2004) was evaluated using real world data in simulated networks of sizes ranging from 1 to 100 nodes; CSM (White et al., 1996) was also evaluated by small-scale simulation of intrusive activities.

4.2. *Alert correlation*

The existing alert correlation techniques used by CIDSs can be classified into four groups: *similarity based*, *attack scenario based*, *multi-stage*, and *filter based*. Similarity based approaches correlate alerts based on the similarity between alert attributes. A function is usually used to calculate the similarity between two pairs of alerts, and the resulting score determines if these alerts will be correlated. Attack scenario based approaches correlate alerts based on predefined attack scenarios. These attack scenarios can either be specified by the users, or learned from training datasets. Multi-stage approaches correlate alerts based on the causality of earlier and later alerts. This approach tries to reconstruct some complex attack scenarios by linking individual steps that are a part of the same attack. Filter based approaches prioritize the alerts by their criticality to the protected systems.

4.2.1. *Similarity based approaches*

Valdes and Skinner (2001) proposed a probabilistic alert correlation approach for the EMERALD (Porras and Neumann, 1997) project. They introduced three levels of sensor correlation: the *synthetic attack threads*[1] *level*, which is used for clustering alerts that are part of the same attack within the detection sensor; the *security incidents level*, which is used for fusing the same attacks reported by multiple sensors; and the *correlated attack reports level*, which is used for merging different steps into a complex attack. In the synthetic attack threads level, alerts are clustered together using a similarity function, to form a single attack thread. The similarity function measures the following features of an alert: the sensor field, attack class, source and target of the alert. In the security incidents level, reports of the same incident from different sensors are fused into a single incident report by using a similarity measure. The features of alerts that are measured in this level are: the attack class, source and target of the attack. In the highest level, the security incidents generated above can be merged to reconstruct various steps in a multi-stage attack by measuring the similarities between the source and target of these incidents.

Debar and Wespi (2001) proposed an aggregation and correlation algorithm for intrusion alerts from various detection sensors. There are two main parts to their aggregation and correlation approach: a unified alert data model and a set of correlation rules. The aggregation and correlation processes are implemented in three steps as follows.

1. Alert preprocessing: the raw alerts from different detection sensors are translated into a unified data model with three attributes – probe (detection sensor), source (source host name or IP address of the alert), and target (destination host name or IP address of the alert). Similar alerts are clustered together based on these three attributes.
2. Relationship correlation: this step correlates alerts that are logically linked with each other, by identifying the *duplicate alerts* and *consequent alerts*. Duplicate alerts are alerts from different detection sensors but having duplicated relationships with each other. Consequent alerts are sets of alerts that are linked in a given order and must occur within a given time interval. The duplicated and alert consequence relationships are defined in a configurable file.
3. Relationship aggregation: alert events generated in the second step are aggregated into seven different *situations*. Situations are different combinations of aggregation attributes, *i.e.*, source, target, and alert class. An alarm will be produced if a threshold on the severity level is satisfied for the situation.

Cuppens (2001) proposed a clustering based method for alert correlation. The correlation process is described as follows. The alerts generated from different IDSs are stored and managed using a relational database. Then the alerts that are mapped to the same occurrence of an attack according to a set of expert similarity rules are grouped into the same cluster. A global alarm is generated for each cluster identified. M2D2 (Morin et al., 2002) is a formal data model for alert correlation. It supplies four types of information to enable complex alert correlation: characteristics of the monitored system, vulnerabilities, security tools for monitoring, and events observed. Seurat (Xie et al., 2004) is a distributed host-based anomaly detection system. It learns normal file update behavior by clustering them across time and space. Accesses that differ from the normal space-time cluster behavior is deemed suspicious. Zhou et al. (2009a) proposed a collaborative intrusion detection approach to detect fast-flux phishing domains. Two similarity based correlation schemes are used to speed-up fast-flux domain detection, i.e., correlating evidence from multiple DNS servers and from multiple suspect fast-flux domains. Spice (Staniford, 2002) and clustering based correlation (Julisch, 2001) also use the similarity based approach for alert correlation. All the alert correlation approaches in this category are effective for clustering similar alerts. However, most of them are limited in their ability to discover the causality between temporary related alerts.

4.2.2. *Attack scenario based approaches*

Complex attacks are usually executed in several steps, where the earlier steps are a preparation for the later attack steps (Valeur, 2006). In order to take this causality into consideration when alerts are being correlated, several attack scenario based approaches have been proposed.

Dain and Cunningham (2001) proposed an algorithm to fuse the alerts from heterogeneous IDSs into attack scenarios by using a probabilistic approach. The proposed CIDS consists of different types of IDSs, which generate alerts separately. These alerts are then converted to a standard format and stored in an SQL database. The fusion system reads from the database to determine to which attack scenario a new alert belongs. Each time a new alert is received from an IDS, it is compared with the attack scenarios being constructed so far. Two probability assignment approaches, one heuristic and one based on a data

[1] Where attacks are detected within a sensor.

mining approach, are proposed to estimate the membership of a new alert. A training data set is used to optimize the parameters of these two probability estimation approaches. A new alert is assigned to the scenario that has the highest probability estimate score. If all the estimate scores are below an assigned threshold, the alert will start a new scenario.

LAMBDA (Cuppens and Ortalo, 2000) is an attack description language that can be used to correlate alerts from different IDSs in a CIDS. LAMBDA defines four components to describe attacks.

1. Pre-condition and post-condition: the condition of the target system that should be satisfied for launching the attack, and the effect on the target system after the attack succeeds.
2. Scenario: a combination of attack events or steps for completing an attack.
3. Detection: a combination of attack events or steps for detecting an attack. This set of events might be different from the scenario since some attack steps are not observable by IDSs.
4. Verification: some conditions on the target system that can verify an attack has succeeded, such as the existence of vulnerabilities in the system.

The first component is defined by a state description language (L_1), which is a form of predicate logic. For example, consider a pre-condition defined by L_1 as "*active_service(s, telnetd)* $\wedge$ *port(telnet, 23, tcp)*". This statement represents that the pre-condition of an attack is that the *telnet* service should be available in the target system on the standard TCP port. Other components are defined by a transition description language (L_2) and an event combining language (L_3). For example, "**scenario**: $expr \in L_3$ **where** $cond \in L_2$ " denotes that if some event transition conditions defined by L_2 are satisfied, then we combine these events into a scenario. The combining process is expressed using L_3. STATL (Eckmann, 2002) is another state-base attack description language that is designed for correlating alerts from different IDSs in a CIDS.

Most alert correlation approaches in this category are effective in detecting some well-documented attacks. However, they fail to detect novel attacks. Furthermore, an explicit attack scenario database can be expensive to build.

4.2.3. *Multi-stage approaches*

In order to address the problem of detecting unknown attacks, several multi-stage alert correlation approaches have been proposed.

Cuppens and Miege (2002) proposed an alert correlation function in a cooperative intrusion detection framework. This approach is based on the assumption that attackers usually perform multiple steps to fulfill their global intrusion plan. The basic idea of this approach is that there are possible logical links between the post-condition of an attack A and the pre-condition of an attack B, *i.e.*, executing a given attack can contribute to executing another attack. All the alerts in the system are modeled using the IDMEF format (Curry and Debar, 2001), and attacks are specified in the LAMBDA language (Cuppens and Ortalo, 2000). Correlation rules are generated based on the attack being described in LAMBDA in an offline fashion. The online correlation process starts after that. When a new alert arrives, it will be compared against historical alerts to check if the correlation conditions are satisfied, which results in a set of correlated alert pairs. Then these pairs will be verified if they belong to any existing attack scenarios. They join an existing scenario if verified, otherwise a new scenario will be started. Abductive correlation is proposed to generate correlation rules online to detect previously undefined attack scenarios.

CAML (Cheung et al., 2003) is a correlated attack modeling language that is designed for detecting multistep attack scenarios. The CAML specification contains a set of modules, where each module comprises three sections, an *activity*, a *pre-condition*, and a *post-condition*. The activity section describes events that trigger this model. The pre-condition section describes the conditions that must be satisfied to trigger this model. The post-condition describes the inference results of this module. All the modules in CAML are logically linked by pre and post-conditions. High-level reusable attack modules are introduced to reduce the cost of attack model development.

Qiu and Lee (Qin, 2005; Qin and Lee, 2003, 2004a,b) proposed a probabilistic inference approach for alert correlation and attack prediction. Isolated alerts are first correlated by using graph techniques, which result in high-level correlation results. Then probabilistic reasoning is used over the results to recognize the attack plan and predict the upcoming attacks. JIGSAW (Templeton and Levitt, 2000) is another attack specification language that is designed for identifying the casual relationship between individual alerts using pre and post-conditions. Ning et al. (2002) proposed a similar alert correlation approach based on the prerequisites and consequences of intrusions. This approach correlates alerts by partially matching the consequence of a historical alert with the prerequisite of a forthcoming alert. Almgren et al. (2008) *presented a correlation model to combine the alerts from several IDSs using different audit sources to improve the overall detection accuracy. A Bayesian network is used to model the detecting sensors and their interdependence. In particular, it analyzes 1) whether an attack is worth investigating, 2) whether a sensor can detect attacks correctly, 3) alerts and other observations from the environment. This model can therefore resolve seemingly conflicting evidence and reason about missing alerts.*

The correlation methods in this category can potentially discover the causal relationship between alerts, and most of them are able to detect unknown attack scenarios. However, these methods often focus on correlated alerts and ignore others that cannot be correlated. The reasoning for discarding these un-correlated alerts has not been rigorously analyzed, and the false alarms generated in the individual IDSs will affect the accuracy of the correlation (Ning and Xu, 2003). Furthermore, a complete library of attack steps is expensive to build considering there are a huge number of attack types.

4.2.4. *Filter based approaches*

In order to remove the need for a complicated attack step library and reduce irrelevant alerts,[2] filter based approaches have been proposed. These approaches prioritize prospective alerts according to their impact to protected systems using a specific filtering algorithm.

[2] Irrelevant alerts are alerts corresponding to attacks that target a non-existent service, *e.g.*, a Windows specific alert can be classified as an irrelevant alert in a Linux system.

M-Correlator (Porras et al., 2002) is an alert correlation prototype that correlates security alerts produced by spatially distributed heterogeneous information security (INFOSEC) devices. It takes into account the topology and operational objectives of the protected network when alerts are being correlated. There are two main processes in M-Correlator: correlation and aggregation. The correlation process involves three phases. Dynamically controllable filters are used in the first phase, which allow the alert producer to unsubscribe from irrelevant alerts. In the second phase, a check is performed against the topology of the target network for incident vulnerability dependencies, then a score of the result is produced. In the third phase, the impact of each alert is prioritized based on (1) the degree of impact of the alert on the critical resources or assets of the target system, and (2) the success probability of the alert. Finally, the aggregation process uses an attribute-based alert clustering algorithm to combine related alerts.

Several approaches have been proposed to use vulnerability analysis to reduce the noise of the alerts generated by individual IDSs in a CIDS, such as Gula (2002) and Kruegel and Robertson (2004). There has also been work on using filters for preventing known vulnerability exploits, such as Vigilante (Costa et al., 2005) and Shield (Wang et al., 2004). Vigilante (Costa et al., 2005) is a collaborative *end-to-end* worm containment system. Each host in the system runs instrumented software to detect worms and broadcast self-certifying alerts (which are machine-verifiable proofs of vulnerability) to all the hosts via a P2P overlay. Alerted hosts then generate filters to block upcoming worm messages. Shield (Wang et al., 2004) is a system for defending against worm attack, by installing some exploit-generic network filters in end systems once a vulnerability is discovered.

Unfortunately, the existing filter based approaches are still at the preliminary stage due to the following limitations.

- The alert correlation methods used in a CIDS need to be deployed in multiple networks with heterogeneous system configurations. However, the filtering algorithms applied in this category are system specific, *i.e.*, alert verification relies on information about the security configuration of the protected network. Consequently, they are expensive to deploy in comparison to the general approaches that support dynamic mechanisms for alert verification.
- The detection accuracy of alert correlation depends on how detailed the description of the patterns that can be found by the filtering algorithm. Consequently, there is a trade-off between the expressiveness of the filtering algorithm and the corresponding computational complexity involved. However, this critical trade-off has not been addressed in existing filter based research.

4.2.5. *Research challenges for alert correlation*

In summary, there are several open issues of existing alert correlation approaches, which are listed as follows.

- How to support increasing levels of expressiveness during correlation, without sacrificing computational efficiency? For example, the similarity based approaches are computationally effective, but they are limited in their ability to discover complicated coordinated attacks due to their lack of alert expressiveness. In contrast, the attack scenario based and multi-stage approaches have sufficient expressiveness to detect complicated coordinated attacks, but their computational complexity and the requirement for complete knowledge of attack behavior make them impractical for use in a large-scale CIDS. The filter based approaches are also expensive to deploy in a large-scale CIDS, since the algorithm needs to be customized to different systems.
- How to maximize the detection accuracy in a CIDS, while minimizing the communication and computational overhead? The attack scenario and multi-stage approaches can achieve a high level of accuracy, assuming that a complete and updated attack type library is in place, but their intensive computational overhead prevents these approaches from promptly detecting attacks in real time. The similarity based and filter based approaches are computationally efficient, but both have limited accuracy, *i.e.*, the similarity based approaches are not able to discover the causality between related alerts, and the filter based approaches are only able to detect system specific attacks.

4.3. *Data privacy*

Data privacy is another important aspect of collaborative intrusion detection. Since participants in a CIDS might come from different organizations, they may be unwilling to share some alerts that contain sensitive information about their network or users. As discussed in the Introduction, while data privacy is outside the main focus of this paper, we summarize the relevant research in this area for completeness.

Lincoln et al. (2004) addressed the problem of privacy concerns in alert correlation. All the IP addresses (both source IP and destination IP) and data captured by IDS sensors are identified as sensitive fields. The configuration and defense coverage of a network site are recognized as sensitive associations. Both sensitive fields and associations might risk being targets of attacks if they are exposed during alert sharing. A set of sanitization techniques are proposed, such as scrubbing sensitive fields, and randomized hot list thresholds.

Xu and Ning (2005) proposed the use of concept hierarchies to balance privacy requirements and the need for intrusion analysis. There are two phases in their approach. First, they use entropy guided alert sanitization to generalize sensitive alert attributes to high-level concepts. Then they define similarity functions between sanitized attributes and build attack scenarios from sanitized alerts.

Gross et al. (2004) proposed a privacy-preserving mechanism using Bloom filters (Bloom, 1970) for use in a CIDS. A central repository in the CIDS receives lists of suspicious IP addresses and sends back alerts on suspicious IP addresses (called a watchlist) to each participant. Bloom filters are employed to encode the watchlist to preserve the privacy of participants. The central repository maintains not only the corresponding Bloom filters for each participant, but also a master Bloom filter for speeding up the lookup process. Locasto et al. (2005) also use a similar Bloom filter based approach to address the issue of privacy-preservation in CIDSs.

4.4. Security and trust

Another important aspect that is outside the main focus of this paper has been the problem of security and trust for collaborative intrusion detection. This issue has been given a much lower priority than other design considerations in CIDSs. There are a few works that have partially addressed this problem in the literature. Several CIDSs (Janakiraman and Zhang, 2003; Yegneswaran et al., 2004) use message authentication to guarantee that alerts come from a trusted participant by using a central certification authority to generate the credentials of the participant. However, this approach cannot protect against a legitimate participant who is sending malicious data. Furthermore, the central certificate authority can become a bottleneck for scalability as the number of participants increases. Chen and Yeager (2001) proposed to build a web of trust between participants. While this approach is promising, there are still issues that need to be addressed, such as how to prevent misbehavior by a peer who has taken the time to first build a high reputation.

4.5. Summary

We summarize in Table 2 the main categories of system architectures and alert correlation algorithms that we have discussed in this paper, in terms of the advantages and disadvantages of

Table 2 – Summary of existing CIDS research.

Main classification		Sample systems	Advantages	Disadvantages
System architecture	Centralized	DIDS (Snapp et al., 1991), DShield (Internet Storm Center), NSTAT (Kemmerer, 1998)	Efficient for small-scale cooperation	Single point of failure; Poor scalability
	Hierarchical	EMERALD (Porras and Neumann, 1997), DSOC (Abdoul Karim Ganame et al., 2008), Li et al. (2007), AAFID (Balasubramaniyan et al., 1998), Servin and Kudenko (2008), NetSTAT (Vigna, 1999)	No single point of failure	Limited scalability; Reduced detection capacity during attacks
	Distributed	Worminator (Locasto et al., 2005), DOMINO (Yegneswaran et al., 2004), Dash et al. (2006), Indra (Janakiraman and Zhang, 2003), MADIDF (Dayong Ye et al., 2008), Garcia et al. (2004), CSM (White et al., 1996)	No single point of failure; Better scalability	Load imbalance during attacks; Uncertain detection accuracy; Simplistic alert correlation
Alert correlation	Similarity Based	Valdes and Skinner (2001), Debar and Wespi (2001), Cuppens (2001), Spice (Staniford, 2002), Zhou et al. (2009a), Seurat (Xie et al., 2004), Julisch (2001)	Easy to implement	Unable to detect complex attacks
	Attack Scenario Based	Dain and Cunningham, 2001, LAMBDA (Cuppens and Ortalo, 2000), STATL (Eckmann, 2002)	Accurate at detecting well-documented attacks	Unable to detect novel attacks; Need to manually define attack conditions
	Multi-stage	Cuppens and Miege, 2002, CAML (Cheung et al., 2003), Qiu and Lee (Qin, 2005; Qin and Lee, 2003, 2004a,b), Almgren et al. (2008), JIGSAW (Templeton and Levitt, 2000), Ning et al. (2002)	Able to detect unknown attacks	Expensive to build complete attack database; Only applies to multi-stage attacks; Expensive to orrelate alerts
	Filter Based	M-Correlator (Porras et al., 2002), Gula (2002), Kruegel and Robertson (2004), Vigilante (Costa et al., 2005), Shield (Wang et al., 2004)	No prior knowledge needed	Lack of generality; Uncertain correlation complexity
Data Privacy		Lincoln et al. (2004), Xu and Ning (2005), Gross et al. (2004), Worminator (Locasto et al., 2005)	Able to preserve the privacy of participants	Loss of alert expressiveness after data sanitization
Security and Trust		Indra (Janakiraman and Zhang, 2003), DOMINO (Yegneswaran et al., 2004), Chen and Yeager (2001)	Able to validate the source of messages	Unable to protect legitimate users from sending malicious data

each approach. While progress has been made on the problem of collaborative intrusion detection, there still remain a number of open problems that need to be addressed, such as:

1. *Expressiveness* – How to balance the trade-off between the expressiveness of the correlation algorithm and corresponding computational complexity during alert correlation in a CIDS.
2. *Scalability* – How to remove the need for a central controller in a CIDS, without sacrificing the overall performance of the CIDS.
3. *Accuracy* – How to improve the detection accuracy of a CIDS, *i.e.*, how to balance the trade-off between the detection rate and false alarm rate in the CIDS.

5. Integrated solutions to collaborative intrusion detection

While there has been considerable research effort into collaborative intrusion detection, three challenges of *expressiveness*, *scalability* and *accuracy* limit the capacity of current collaborative intrusion approaches to detect coordinated attacks. In this section, we highlight the opportunities for a more integrated collaborative intrusion detection approach to address these research problems. Note that there are other issues raised by the design of a CIDS, such as trust, security and privacy among participant IDSs. These issues have already been the subject of research in the distributed computing community. In this section, we consider the case of a CIDS in which the participant IDSs are under the control of reliable network operators on a secure platform. This is often the case within a carrier's network, or between network carriers. The problem of how to operate a CIDS when these assumptions of trust and security are relaxed, *i.e.*, when an IDS may be under the control of an attacker, are outside the main focus of this paper.

A promising approach to the problem of collaborative intrusion detection is via a *content-based correlation* scheme for message communication, *i.e.*, a *publish-subscribe* model for alert correlation. Publish-subscribe models have been widely used in the literature for tasks such as event notification, mobility support services and in the Java Message Service. In the context of collaborative intrusion detection, when a participant IDS detects a possible attack in its monitored subnetwork, it generates an alert, which is reported to the CIDS. This is known as *subscription*, *i.e.*, the IDS is registering its interest to the CIDS to confirm whether the alert is part of a large-scale coordinated attack. The role of the CIDS is to correlate alerts that are subscribed by participating IDSs. If enough subscribed alerts are received to confirm an attack, then the CIDS *publishes* a *notification* of a confirmed attack to the participating IDSs that subscribed to the attack.

There are two complementary algorithms we have considered for alert correlation in the above collaborative intrusion detection model (Zhou et al., 2009b): *single-feature correlation*, where alerts are correlated on the basis of a single traffic feature, such as the source IP address of the suspicious traffic; and *multi-dimensional alert correlation*, where patterns of alerts can be found based on multiple traffic features.

In general, there are three collaboration models to distributing computation in a CIDS, as shown in Fig. 5. The first approach is centralized collaboration (Fig. 5(a)), in which all correlation is performed at a centralized node. Alerts are subscribed to the centralized node by participating IDSs. All alerts are correlated at the centralized node, which notifies the relevant IDSs of any confirmed attacks. Compared to the other models, this centralized collaboration architecture has the highest overall accuracy, as all information is available at a single location. One of the key research problems is how to find a trade-off between the sensitivity and false alarm rate in such a CIDS. In Zhou et al. (2007), we proposed an optimization scheme for the key parameters in this CIDS model, with an empirical evaluation of this scheme using alerts from a global repository of IDS logs for both stealthy scans and worm outbreaks.

The second approach is single-level hierarchical collaboration, as shown in Fig. 5(b). In this approach, some correlation can be performed locally by the participating IDSs, so that not all alerts need to be subscribed to the central correlation node. This can reduce the computational load on the centralized node, in order to support more sophisticated algorithms that can be used to find more expressive (*i.e.*, computationally expensive) patterns of alerts. Our results have shown that this two-stage hierarchical scheme achieves a significant reduction in alert messages at the global stage with little degradation in detection accuracy in a variety of attack scenarios.

The third approach is to eliminate the need for a centralized correlation node, so that the correlation load can be

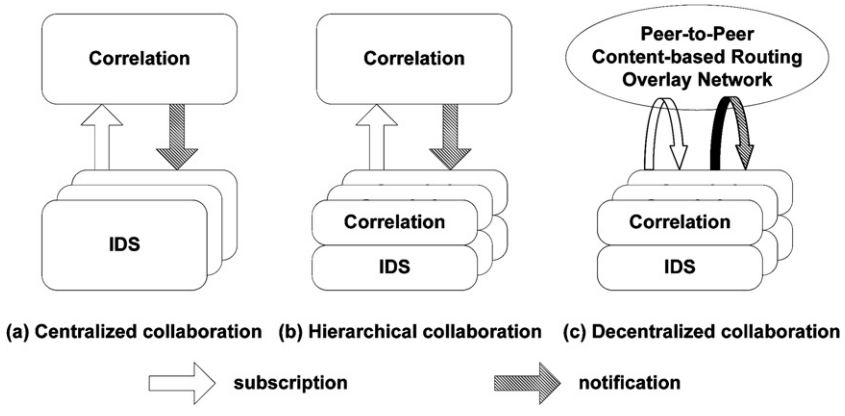

Fig. 5 – Three types of collaborative intrusion detection models.

distributed between the participating IDSs in a decentralized manner. In particular, a peer-to-peer (P2P) communication scheme is supported in this approach. For this to work in a scalable manner, a method is required to route subscribed alerts automatically to the responsible peer for correlation, so that peers do not need to keep track of which peers are responsible for which attack instances. It is achieved by using a P2P content-based routing overlay network between the participating IDSs (Zhou et al., 2005), as shown in Fig. 5(c). This raises the question of how to map the alert correlation task into this P2P overlay. While the elimination of a central node for correlation enables greater scalability in terms of distributing the computational load, the content-based routing overlay network introduces a routing delay to the system. Our results have shown that in practice, the reduction in correlation delay outweighs the increase in communication and routing delay on wide-area networks. However, a potential shortcoming of this approach is that under conditions that produce a focused load, individual nodes may become special cases of the centralized model. As we demonstrated in (Zhou et al., 2008), a load ''hot-spot'' can be created during a worm outbreak. In addition, participants in the system may be uncomfortable with storing their raw alert information at a single node if there is no guaranteed trust between participants. Of relevance to this problem, an alternative model of distributed alert correlation is to use a correlation group scheduling technique to periodically reshuffle correlation groups among peers, with the aim of maximizing bandwidth savings and minimizing detection delay (Locasto et al., 2005). A related approach has been proposed that uses a gossip protocol to propagate alerts to a random subset of correlation nodes (Dash et al., 2006).

6. Conclusion

Coordinated attacks are a widespread problem, and extremely difficult to detect, since the evidence of suspicious activities is spread across multiple network domains. However, the common attack topology of these attacks sheds light on an approach for detection, *i.e.*, there is a common stage in which all the attack traffic is either coming from the same source (in the case of large-scale stealthy scans and worm outbreaks) or going to the same destination (in the case of DDoS attacks). CIDSs are a promising approach to meet the detection challenge of these coordinated attacks by correlating suspicious evidence from different IDSs. We have presented an overview of the state-of-the-art in collaborative intrusion detection research. We have classified CIDSs into different categories based on the system architecture they adopt, and the alert correlation algorithm they use for alert analysis in their systems. Based on this survey, we have identified three open research problems for detecting large-scale coordinated attacks, namely: expressiveness, scalability and accuracy. Initial progress towards these challenges has demonstrated the feasibility of collaborative intrusion detection over large-scale deployments on the Internet. By combining the resources of multiple intrusion detection systems, there is an emerging hope that we can combat the growing sophistication of attacks on the Internet.

Acknowledgment

This research was supported by the Australian Research Council.

REFERENCES

Abdoul Karim Ganame RB, Bourgeoisa Julien, Spiesa F. A global security architecture for intrusion detection on computer networks. Computers & Security 2008;27:30–47.

Almgren M, Lindqvist U, Jonsson E. A multi-sensor model to improve automated attack detection. In: Proceedings of the 11th international symposium on recent advances in intrusion detection. Springer; 2008. p. 291–310.

Balasubramaniyan J, Garcia-Fernandez J, Isacoff D, Spafford E, Zamboni D. An architecture for intrusion detection using autonomous agents. In: Proceedings of the 14th IEEE computer security applications conference; 1998. p. 13–24.

Barford VYP, Jha S. Fusion and filtering in distributed intrusion detection systems. In: Proceedings of annual allerton conference on communication, control and computing; September 2004.

Bloom BH. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM 1970;13(7):422–6.

Brenner B. Sasser shows there must be a better way, http://searchsecurity.techtarget.com/originalContent/0,289142,sid14_gci963170,00.html; 2004.

Brugger ST. Data mining methods for network intrusion detection. Tech. Rep.. Davis: University of California; 2004

CERT Coordination Center (CERT/CC). CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks, http://www.cert.org/advisories/CA-1996-21.html; 1996.

CERT Coordination Center (CERT/CC). CERT incident note IN-2001-09, http://www.cert.org/incident_notes/IN-2001-09.html; 2001.

CERT Coordination Center (CERT/CC). CERT advisory CA-2003-04 MS-SQL server worm, http://www.cert.org/advisories/CA-2003-04.html; 2003a.

CERT Coordination center (CERT/CC). Module 4-types of intruder attacks; 2003b.

CERT Coordination Center (CERT/CC). US-CERT current activity archive, http://www.cert.org/current/archive/2004/07/12/archive.html; 2004.

CERT Coordination Center (CERT/CC). CERT/CC statistics 1988–2006, http://www.cert.org/stats; 2006.

Chen R, Yeager W. Poblano: a distributed trust model for peer-to-peer networks. JXTA Security Project White Paper; 2001. p. 1–26.

Cheung S, Lindqvist U, Fong MW. Modeling multistep cyber attacks for scenario recognition. In: Proceedings of the third DARPA information survivability conference and exposition (DISCEX); 2003. p. 284–92.

Chiueh T. Constraints, style and focus of industrial security research. In: Keynote in the 10th international symposium on Recent Advances in Intrusion Detection (RAID); 2007.

Costa M, Crowcroft J, Castro M, Rowstron A, Zhou L, Zhang L, et al. Vigilante: end-to-end containment of internet worms. In: Proceedings of the twentieth ACM symposium on operating systems principles (SOSP 05); 2005. p. 133–47.

Cuppens F. Managing alerts in a multi-intrusion detection environment. In: Proceedings of the 17th annual computer security applications conference (ACSAC); 2001. p. 22–31.

Cuppens F, Miege A. Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the 2002 IEEE symposium on security and privacy (SP); 2002. p. 202–15.

Cuppens F, Ortalo R. Lambda: a language to model a database for detection of attacks. In: Proceedings of recent advances in intrusion detection (RAID); 2000. p. 197–16.

Curry D, Debar H. Intrusion detection message exchange format data model and extensible markup language (XML) document type definition: draft-itetfidwg-idmef-xml-03.txt, http://www.ietf.org; February 2001.

Dain O, Cunningham R. Fusing a heterogeneous alert stream into scenarios. In: Proceedings of the 2001 ACM workshop on data mining for security applications; 2001. p. 1–13.

Dash D, Kveton B, Agosta J, Schooler E, Chandrashekar J, Bachrach A, et al. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In: Proceedings of the twenty-first national conference on artificial intelligence (AAAI); 2006. p. 1115–22.

Dayong Ye MZ, Quan Bai, Ye Z. P2P distributed intrusion detections by using mobile agents. In: Seventh IEEE/ACIS international conference on computer and information science, 2008 (ICIS 08); May 2008. p. 259–65.

Debar H, Wespi A. Aggregation and correlation of intrusion-detection alerts. In: Proceedings of the 4th international symposium on recent advances in intrusion detection (RAID); 2001. p. 85–103.

Dietrich S, Long N, Dittrich D. Analyzing distributed denial of service tools: the shaft case. In: Proceedings of USENIX LISA; 2000. p. 329–39.

Eckmann S. Statl: an attack language for state-based intrusion detection. Journal of Computer Security 2002;10(1):71–103.

Franklin J, Paxson V, Perrig A, Savage S. An inquiry into the nature and causes of the wealth of Internet miscreants. In: Proceedings of the ACM conference on computer and communications security (CCS); 2007.

Garber L. Denial-of-service attacks rip the Internet. IEEE Computer 2000;33(4):12–7.

Garcia J, Autrel F, Borrell J, Castillo S, Cuppens F, Navarro G. Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation. In: Sixth international conference on information and communications security; October, 2004. p. 223–35.

Goldi C, Hiestand R. Scan Detection Based Identification of Worm-Infected Hosts. ETHZ, Zurich: Swiss Federal Institute of Technology; April 2005. master's Thesis

Gross P, Parekh J, Kaiser G. Secure selecticast for collaborative intrusion detection systems. In: Proceedings of the 3rd international workshop on distributed event-based systems (DEBS); 2004.

Gula R. Correlating IDS alerts with vulnerability information. Tech. Rep. Tenable Network Security; December 2002. Special Publication 800–831.

Heberlein LT, Mukherjee B, Levitt KN. Internetwork security monitor: an intrusion-detection system for large-scale networks. In: Proceedings of the 15th national computer security conference; 1992. p. 262–71.

Hochberg J, Jackson K, Stallings C, McClary JF, DuBois D, Ford J. Nadir: an automated system for detecting network intrusion and misuse. In: Proceedings of the 15th national computer security conference; 1993. p. 235–48.

Huang M, Jasper R, Wicks T. A large scale distributed intrusion detection framework based on attack strategy analysis. Computer Networks 1999;31(23–24):2465–75.

Igure VM, Williams RD. Taxonomies of attacks and vulnerabilities in computer systems. IEEE Communications Surveys & Tutorials 2008;10(1):6–19.

Internet Storm Center. DShield.org, http://www.dshield.org.

Janakiraman R, Zhang M. Indra: a peer-to-peer approach to network intrusion detection and prevention. In: Proceedings of the twelfth IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises (WETICE); 2003. p. 226–31.

Julisch K. Mining alarm clusters to improve alarm handling efficiency. In: Proceedings of the 17th annual computer security applications conference (ACSAC); 2001. p. 12–21.

Jung J, Paxson V, Berger A, Balakrishnan H. Fast portscan detection using sequential hypothesis testing. In: Proceedings of the IEEE symposium on security and privacy; 2004. p. 211–25.

Kemmerer RA. NSTAT: a model-based real-time network intrusion detection system. University of California at Santa Barbara; 1998. Tech. Rep. TRCS97-18.

Kruegel C. Internet security. The Industrial Communication Technology Handbook; February 2005.

Kruegel C, Robertson W. Alert verification: determining the success of intrusion attempts. In: Proceedings of the first workshop the detection of intrusions and malware and vulnerability assessment (DIMVA); 2004.

Lakhina A, Crovella M, Diot C. Diagnosing network-wide traffic anomalies. ACM SIGCOMM Computer Communication Review 2004;34(4):219–30.

Lakhina A, Crovella M, Diot C. Mining anomalies using traffic feature distributions. In: Proceedings of the 2005 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM); 2005. p. 217–28.

Lee W, Stolfo SJ. A framework for constructing features and models for intrusion detection systems. Information and System Security 2000;3(4):227–61.

Li J, Lim D, Sollins K. Dependency-based distributed intrusion detection. In: Proceedings of the DETER community workshop on cyber security experimentation and test. CA, USA: USENIX Association Berkeley; 2007.

Lincoln P, Porras P, Shmatikov V. Privacy-preserving sharing and correlation of security alerts. In: Proceedings of the 13th USENIX security symposium; August 2004. p. 239–54.

Locasto M, Parekh J, Stolfo S, Keromytis A, Malkin T, Misra V. Collaborative distributed intrusion detection. Dept. Computer Science, Columbia Univ.; 2004. Tech. Rep. CUCS-012-04, Tech. Rep.

Locasto M, Parekh J, Keromytis A, Stolfo S. Towards collaborative security and P2P intrusion detection. In: Proceedings of the 2005 IEEE workshop on information assurance and security; 2005. p. 333–39.

McPherson Danny. 2% of Internet traffic raw sewage, http://asert.arbornetworks.com/2008/03/2-of-internet-traffic-raw-sewage; 2008.

Miller C. The legitimate vulnerability market: Inside the secretive world of 0-day exploit sales. In: Proceedings of the sixth workshop on the economics of information security; 2007.

Mirkovic J, Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review 2004;34(2):39–53.

Moore D, Shannon C, Brown J. Code Red: a case study on the spread and victims of an Internet worm. In: Proceedings of the 2002 ACM SICGOMM Internet Measurement Workshop; 2002. p. 273–84.

Moore D, Paxson V, Savage S, Shannon C, Staniford S, Weaver N. Inside the slammer worm. IEEE Security & Privacy Magazine 2003;1(4):33–9.

Morin B, Me L, Debar H, Ducasse M. M2D2: a formal data model for IDS alert correlation. In: Proceedings of the 5th international symposium on recent advances in intrusion detection (RAID); 2002. p. 115–37.

Mounji A, Le Charlier B, Zampunieris D, Habra N. Distributed audit trail analysis. In: Proceedings of the internet society symposium on network and distributed system security (ISOC); February 1995. p. 102–13.

Nichols Shaun. Storm worm seeks out April fools, http://www.itnews.com.au/News/73086,storm-worm-seeks-out-april-fools.aspx; 2008.
Ning P, Xu D. Learning attack strategies from intrusion alerts. In: Proceedings of the 10th ACM conference on computer and communications security (CCS); 2003. p. 200–09.
Ning P, Cui Y, Reeves DS. Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM conference on computer and communications security (CCS); 2002. p. 245–54.
Paxson V. An analysis of using reflectors for distributed denial-of-service attacks. ACM SIGCOMM Computer Communication Review 2001;31(3):38–47.
Peng T, Leckie C, Ramamohanarao K. Survey of network-based defense mechanisms countering the dos and ddos problems. ACM Transactions on Computational Logic 2006;2(3):1–35.
Porras P, Neumann P. Emerald: event monitoring enabling responses to anomalous live disturbances. In: Proceedings of the 20th national information systems security conference; 1997. p. 353–65.
Porras PA, Fong MW, Valdes A. A mission-impact-based approach to INFOSEC alarm correlation. In: Proceedings of recent advances in intrusion detection (RAID); 2002. p. 95–114.
Qin X. A probabilistic-based framework for infosec alert correlation. Ph.D. dissertation. Atlanta, GA, USA: Georgia Institute of Technology; 2005.
Qin X, Lee W. Statistical causality analysis of infosec alert data. In: Proceedings of recent advances in intrusion detection (RAID); 2003. p. 73–93.
Qin X, Lee W. Attack plan recognition and prediction using causal networks. In: Proceedings of the 20th annual computer security applications conference (ACSAC); 2004a. p. 370–79.
Qin X, Lee W. Discovering novel attack strategies from infosec alerts. In: Proceedings of the 9th European symposium on research in computer security (ESORICS); 2004b.
Rabiner L. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 1989;77(2):257–86.
Savage S. Internet outbreaks: epidemiology and defenses. In: Invided talk in the 12th annual network and distributed system security symposium (NDSS); 2005.
Servin A, Kudenko D. Multi-agent reinforcement learning for intrusion detection, lecture notes in computer science, vol. 4865; 2008. p. 211–23.
Snapp S, Brentano J, Dias G, Goan T, Heberlein L, Ho C, et al. DIDS (distributed intrusion detection system) – motivation, architecture, and an early prototype. In: Proceedings of the 14th national computer security conference; 1991. p. 167–76.
Staniford S. Practical automated detection of stealthy portscans. Journal of Computer Security 2002;10(1):105–36.
Staniford S, Moore D, Paxson V, Weaver N. The top speed of flash worms. In: Proceedings of the 2004 ACM workshop on rapid malcode; 2004. p. 33–42.
Staniford-Chen S, Cheung S, Crawford R, Dilger M, Frank J, Hoagland J, et al. Grids-a graph based intrusion detection system for large networks. In: Proceedings of the 19th national information systems security conference. vol. 1; September 1996. p. 361–70.
Stolfo SJ, Lee W, Chan PK, Fan W, Eskin E. Data mining-based intrusion detectors: an overview of the Columbia ids project. ACM SIGMOD Record 2001;30(4):5–14.
Symantec Threat Advisory Center. Outbreak alert: storm trojan, http://www.symantec.com/outbreak/storm_trojan.html; 2007.
Templeton S, Levitt K. A requires/provides model for computer attacks. In: Proceedings of new security paradigms workshop; 2000. p. 31–38.
Valdes A, Skinner K. Probabilistic alert correlation. In: Proceedings of the 4th international symposium on recent advances in intrusion detection (RAID); 2001. p. 54–68.
Valeur F. Real-time intrusion detection alert correlation. Ph.D. dissertation. Santa Barbara: University of California; May 2006.
Vigna G. Netstat: a network-based intrusion detection system. Journal of Computer Security 1999;7(1):37–71.
Wang HJ, Guo C, Simon DR, Zugenmaier A. Shield: vulnerability-driven network filters for preventing known vulnerability exploits. In: Proceedings of the 2004 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM); 2004. p. 193–04.
Weaver N, Paxson V, Staniford S, Cunningham R. A taxonomy of computer worms. In: Proceedings of the 2003 ACM workshop on rapid malcode. 2003. p. 11–18.
White GB, Fisch EA, Pooch UW. Cooperating security managers: a peer-based intrusion detection system. IEEE Network 1996; 10(1):20–3.
Xie Y, Kim H, O Hallaron D, Reiter M, Zhang H. Seurat: a pointillist approach to anomaly detection. In: Proceedings of the 7th international symposium on Recent Advances in Intrusion Detection (RAID). Springer; 2004. p. 238–57.
Xu D, Ning P. Privacy-preserving alert correlation: A concept hierarchy based approach. In: Proceedings of the 21st annual computer security applications conference (ACSAC); December 2005. p. 489–98.
Yegneswaran V, Barford P, Jha S. Global intrusion detection in the DOMINO overlay system. In: Proceedings of network and distributed security symposium (NDSS); 2004.
Zhou CV, Karunasekera S, Leckie C. A peer-to-peer collaborative intrusion detection system. In: Proceedings of the IEEE international conference on networks (icon). Malaysia; 2005. pp. 118–23.
Zhou CV, Karunasekera S, Leckie C. Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In: Proceedings of the tenth IFIP/IEEE international symposium on integrated network management (IM). Germany; 2007. p. 80–9.
Zhou CV, Karunasekera S, Leckie C. Relieving hot spots in collaborative intrusion detection systems during worm outbreaks. In: The 11th IEEE/IFIP network operations and management symposium (NOMS 2008); April 2008. p. 49–6.
Zhou CV, Leckie C, Karunasekera S. Collaborative detection of fast flux phishing domains. Journal of Networks 2009a;4:75–84.
Zhou CV, Leckie C, Karunasekera S. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. Journal of Network and Computer Applications February 2009b.

Chenfeng Vincent Zhou received his Ph.D. in computer science from the University of Melbourne, Australia, in 2009. He is currently a Research Fellow at The University of Melbourne. He is a Certified Information Systems Security Professional (CISSP) since 2008. His research interests include computer security, network management and distributed systems.

Christopher Leckie is an Associate Professor in the Department of Computer Science and Software Engineering at the University of Melbourne, Australia. His research interests include using data mining and other artificial intelligence techniques for network intrusion detection and network management, as well as the management of sensor networks. Prior to joining the University of Melbourne, he was a Principal Engineer at Telstra Research Laboratories, where he

conducted research and development into artificial intelligence techniques for various telecommunication applications.

Shanika Karunasekera received the B.Sc (Honours) degree in electronics and telecommunications engineering from the University of Moratuwa, Sri Lanka, in 1990 and the Ph.D. degree in electrical engineering from the University of Cambridge, UK, in 1995. From 1995 to 2002, she was a Software Engineer and a Distinguished Member of Technical Staff at Lucent Technologies, Bell Labs Innovations, USA. Since January 2003, she has been a Senior Lecturer at the Department of Computer Science and Software Engineering, University of Melbourne. Her current research interests are distributed computing, software engineering and peer-to-peer computing. Dr Karunasekera is a member of the ACM.