ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/276205397

Taxonomy and Survey of Collaborative Intrusion Detection

Article *in* ACM Computing Surveys · May 2015

DOI: 10.1145/2716260

CITATIONS
164

READS
3,780

4 authors:

Emmanouil Vasilomanolakis
Aalborg University
41 PUBLICATIONS **511** CITATIONS
SEE PROFILE

Shankar Karuppayah
Universiti Sains Malaysia
38 PUBLICATIONS **437** CITATIONS
SEE PROFILE

Max Mühlhäuser
Technische Universität Darmstadt
633 PUBLICATIONS **4,775** CITATIONS
SEE PROFILE

Mathias Fischer
University of Hamburg
64 PUBLICATIONS **468** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

CRC MAKI - Multi-Mechanisms Adaptation for the Future Internet (2017- at least 2020) View project

IMMUNE View project

All content following this page was uploaded by Mathias Fischer on 13 October 2015.

The user has requested enhancement of the downloaded file.

Taxonomy and Survey of Collaborative Intrusion Detection

EMMANOUIL VASILOMANOLAKIS, SHANKAR KARUPPAYAH, MAX MÜHLHÄUSER, and MATHIAS FISCHER, Technische Universität Darmstadt/CASED

The dependency of our society on networked computers has become frightening: In the economy, all-digital networks have turned from facilitators to drivers; as cyber-physical systems are coming of age, computer networks are now becoming the central nervous systems of our physical world—even of highly critical infrastructures such as the power grid. At the same time, the 24/7 availability and correct functioning of networked computers has become much more threatened: The number of sophisticated and highly tailored attacks on IT systems has significantly increased. Intrusion Detection Systems (IDSs) are a key component of the corresponding defense measures; they have been extensively studied and utilized in the past. Since conventional IDSs are not scalable to big company networks and beyond, nor to massively parallel attacks, Collaborative IDSs (CIDSs) have emerged. They consist of several monitoring components that collect and exchange data. Depending on the specific CIDS architecture, central or distributed analysis components mine the gathered data to identify attacks. Resulting alerts are correlated among multiple monitors in order to create a holistic view of the network monitored. This article first determines relevant requirements for CIDSs; it then differentiates distinct building blocks as a basis for introducing a CIDS design space and for discussing it with respect to requirements. Based on this design space, attacks that evade CIDSs and attacks on the availability of the CIDSs themselves are discussed. The entire framework of requirements, building blocks, and attacks as introduced is then used for a comprehensive analysis of the state of the art in collaborative intrusion detection, including a detailed survey and comparison of specific CIDS approaches.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: Security and Protection

General Terms: Security

Additional Key Words and Phrases: Collaborative intrusion detection, attacks, classification, network security

ACM Reference Format:
Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 2015. Taxonomy and survey of collaborative intrusion detection. ACM Comput. Surv. 47, 4, Article 55 (May 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2716260

1. INTRODUCTION

The Internet encompasses nearly every aspect of our lives today and has thus evolved into a critical infrastructure with severe consequences in the case of partial and temporal failure. At the same time, the number of sophisticated and specifically tailored attacks on connected systems has increased considerably [Sood and Enbody 2013].

This work was supported by AGT International.
Authors' addresses: E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, Telecooperation Group, Technische Universität Darmstadt/Center for Advanced Security Research Darmstadt (CASED); emails: {emmanouil.vasilomanolakis, shankar.karuppayah}@cased.de, max@informatik.tu-darmstadt.de, mathias.fischer@ieee.org.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2015 ACM 0360-0300/2015/05-ART55 $15.00
DOI: http://dx.doi.org/10.1145/2716260

Moreover, a large number of unreported or even completely unnoticed intrusion cases may exist. One of the lessons we had to learn with the rise of the Internet is that we cannot protect ourselves from all kind of possible attacks. IT systems developed by humans cannot be totally secure.

For this reason, in addition to implementing software and designing hardware to be as secure as possible, it is inevitable that IT systems must be continuously monitored to ensure their correct functioning, for any kind of anomalies, or for signs of intrusions and attacks. Thus, the monitoring process provides a second line of defense for any kind of (critical) network infrastructure and IT systems. Such a task is usually taken over by Intrusion Detection Systems (IDSs). An IDS monitors a host or a network and analyzes it for signs of intrusions manifested by malicious behavior or security policy violations. Thus, its goal is the detection of any attempt to compromise confidentiality, integrity, availability, or simply to bypass the security mechanisms of a computer or network [Barry and Chan 2010].

In regards to the position of their deployment, IDSs can be divided into either host-based or network-based IDSs [Kabiri and Ghorbani 2005; Lazarevic et al. 2005]. Host-based IDSs analyze events and the behavior of users on the granularity of single devices. This allows us to collect detailed information, but introduces additional computational overhead that can affect the overall performance of the monitored system. Moreover, this requires a deployment of IDSs on all devices to be protected. In contrast, network-based IDSs can protect several devices or even entire networks at once because they monitor only network traffic.

IDSs can be further categorized according to their deployed detection mechanisms into signature-based and anomaly-based. Signature-based IDSs search for signatures of known attacks and detect their occurrence in the network. Anomaly-based IDSs attempt to initially learn the normal system state and afterward define any deviating behavior as an intrusion [Axelsson 2000; Chandola et al. 2009]. In contrast to a signature-based detection, an anomaly-based detection can also detect unknown attacks. However, usually, this comes at the cost of a high *false-positive* rate, whereas a signature-based detection usually results in more *false negatives*.

In addition, IDSs can be based either on completely passive monitoring, or they can also employ more active components such as *honeypots*. Honeypots are systems whose value lies in being probed, attacked, or compromised [Spitzner 2003]. Their usage can increase the overall accuracy of the IDS [Kabiri and Ghorbani 2005] because, by definition, they exhibit a zero false-positive ratio.

The early IDSs have been mostly isolated single instances for monitoring a single system or a single network by carrying out local analysis for attacks. Hence, between instances of such a stand-alone IDS, no communication and interaction takes place. Obviously, such a solution will not detect sophisticated and highly distributed attacks. Isolated IDSs will not be able to establish connections between malicious events occurring at different places at the same time. Thus, for the protection of large networks and large IT ecosystems, *Collaborative IDSs (CIDSs)* emerged. CIDSs consist of several monitors that act as sensors and collect data. Furthermore, they usually contain one or several analysis units carrying out the actual intrusion detection on the data obtained from the sensors. In addition to improving the detection accuracy of a monitored network, CIDSs can also significantly reduce the complex tasks of security administrators [Goodall et al. 2004]. Depending on the specific CIDS, monitors and analysis units can be also co-located. If not noted otherwise and for the remainder of this article, we assume that a monitor is a combination of both a sensor and an analysis unit. CIDSs enforce cooperation among different monitors and thus are more scalable than stand-alone IDSs. CIDSs can be classified according to their communication architecture, as shown in Figure 1, into centralized, decentralized, and distributed CIDS:

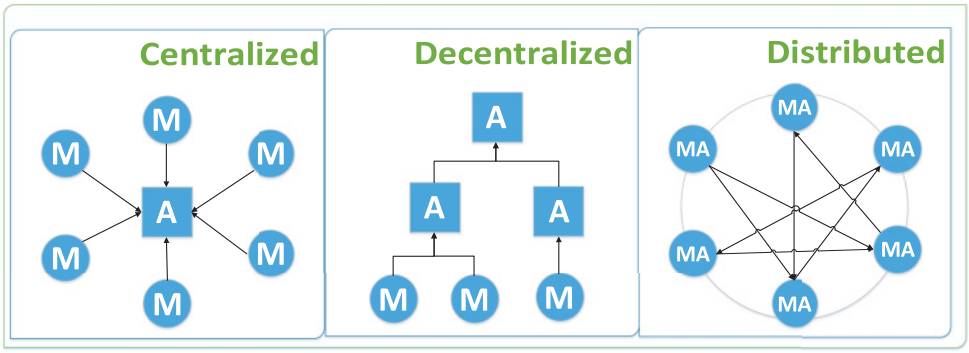

Fig. 1. Overview of centralized, decentralized, and distributed IDS architectures that consist of monitors (*M*) and analysis units (*A*).

—*Centralized CIDSs* consist of several monitors that monitor the behavior of their respective host or the network traffic passing by. These monitors share their data with a *central analysis unit*; these data can be either alerts as a result of a local detection or extracted data from the local network traffic. Hence, the analysis unit is either applying alert correlation algorithms on top of received alerts or standard detection algorithms on top of received network traffic data. Similar to isolated IDSs, centralized CIDSs do not scale with the increasing size of the system that needs to be protected. Moreover, the central analysis unit represents a performance bottleneck and a Single Point of Failure (SPoF).
—*Decentralized CIDSs* usually employ a hierarchical structure of monitoring points or multiple self-contained IDS deployments. Through this structure, they overcome the performance bottleneck of centralized CIDSs because they employ preprocessing and correlation of the monitored data within the hierarchy until the data converge to a central analysis unit on top.
—*Distributed CIDSs* share the tasks of the central analysis unit equally among all monitors, so that each monitor is also an analysis unit. Thus, distributed CIDSs usually employ a Peer-to-Peer (P2P) architecture, in which monitored data are correlated, aggregated, and analyzed in a completely distributed manner among the monitors.

Despite the fact that a lot of work has been done in surveying and classifying IDSs [Debar et al. 1999; Axelsson 2000; Chandola et al. 2009; Barry and Chan 2010], only a few focus specifically on CIDSs and their architecture [Bye et al. 2010; Zhou et al. 2010]. In particular, Zhou et al. [2010] presented a study of CIDSs and focused mainly on the system architectures and the correlation of alert data. Their work differs from ours in that they only briefly describe a few examples of existing CIDS proposals per architecture. Hence, their focus was not on giving a detailed survey of CIDSs proposals, nor it was on discussing attacks for CIDS evasion.

In this article, we define requirements for the successful deployment of CIDSs that are intended for the protection of large IT systems and critical infrastructures. We disassemble and reduce CIDSs to their basic building blocks and extensively discuss the design space of each of them. On the basis of the requirements and identified building blocks, we summarize attacks for CIDS evasion and attacks on the availability of CIDSs themselves. Finally, the main contribution of this article is a detailed survey of current CIDSs following the classification into centralized, decentralized, and distributed

CIDSs. All solutions presented in this article are extensively discussed along with defined requirements and building blocks, as well as the possible attacks.

The remainder of this article is organized as follows: In Section 2, we define the requirements of CIDSs and subsequently present attacks on and evasion techniques of CIDSs based on the scope of the attacker. Section 3 presents the design space for CIDSs and proposes a disjunction of CIDSs into five main build blocks. In Sections 4 through 6, we present a survey of CIDSs and discuss all individual approaches with respect to our requirements, the identified attacks, and the different CIDS building blocks. In Section 7, we provide a detailed comparison of the discussed CIDSs. Finally, Section 8 concludes the article and summarizes research gaps and necessary future work.

2. REQUIREMENTS FOR CIDSs AND CORRESPONDING ATTACKS

In this section, we first summarize the requirements of CIDSs for deployment in large networks and IT systems. Thereafter, we discuss attacks on CIDSs and how they affect the requirements of CIDSs.

2.1. Requirements

A CIDS for protecting large networks or large IT systems has to fulfill the following requirements; these have been partially introduced in our former work [Vasilomanolakis et al. 2013]:

—**Accuracy:** Accuracy for IDSs is determined by the percentage of successfully detected attacks and the corresponding percentage of undetected attacks (*false negatives*). In addition, the number of falsely triggered alarms (*false positives*) also needs to be taken into account to measure the accuracy of an IDS. An accurate IDS should minimize both.
—**Minimal overhead:** Overhead arises in terms of *computation* and *communication* effort. The techniques used to produce, collect, or correlate intrusion alerts must have a low computational overhead. In addition, signaling inside the IDS (e.g., between monitors, or between monitors and an analysis units) needs to be as minimal as possible.
—**Scalability:** Scalability requires that the performance of the IDS increases linearly with the size of the resources added, so that networks of arbitrary size can be protected [Hill 1990]. Therefore, the IDS should not contain bottlenecks or SPoF.
—**Resilience:** In the presence of failures in internal components and during attacks, a CIDS should still maintain its availability and ensure an acceptable accuracy. Hence, a CIDS should not only be resilient to system malfunctions and external attacks like Denial of Service (DoS), but also to internal attacks from malicious CIDS components and malicious systems in the protected network/system. For this reason, a CIDS should prevent Single Point of Failures (SPoFs) and should provide graceful degradation and fast restoration mechanisms to counter failures and attacks.
—**Privacy:** In a collaborative environment, exchanged alerts may include sensitive information that needs to be protected and should not be shared or disclosed with all components in a CIDS. This is particularly important for CIDS deployments that share data across domains that require privacy protection for the involved users, companies, and network providers.
—**Self-configuration** is the ability of the system to automatically adjust itself, without the intervention of an administrator. In contrast to systems that require manual configuration, this provides the ability of constructing less error-prone systems.
—**Interoperability** is the ability of the system to interoperate with instances of the same system deployed in other networks and also across different IDS

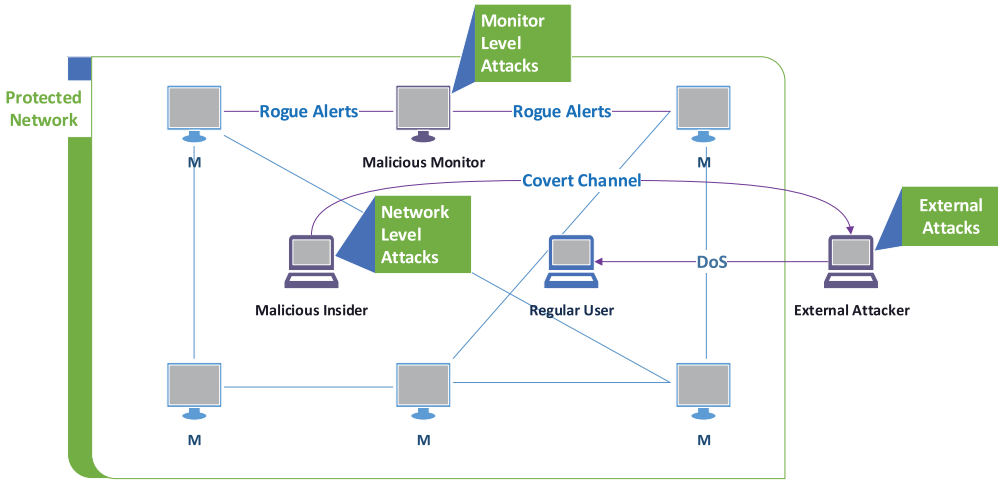

Fig. 2. Possible network positions of attackers. *M* represents the different *monitoring points* of the CIDS.

implementations. For instance, this can be achieved via the utilization of standardized formats such as the Intrusion Detection Message Exchange Format (IDMEF).

2.2. Attacks on CIDS

CIDSs are basically IT systems and thus they can be targets of attacks. Therefore, before designing or analyzing such a system, it is essential to be aware of all the possible weak links and vulnerabilities that these systems might exhibit.

Attacks on CIDSs can be classified into internal and external attacks based on the scope of the attackers. In an external attack, the adversary may try to detect the presence of a CIDS, launch evasion attacks, or attack specific components of the IDS directly, for example by degrading its service availability via a Distributed Denial of Service (DDoS) attack [Srivastava et al. 2011]. Internal attacks refer to malicious behavior originated from within the monitored network. Thus, either a host within the monitored network (network level) or a monitor that is part of the CIDS (monitor level) has been compromised.

The different positions of the adversary with respect to the aforementioned classifications are shown in Figure 2. We consider three different network positions: an *external attacker* (e.g., carrying out DDoS attacks), a *malicious insider* (e.g., performing covert channel attacks), and a *malicious monitoring point* (e.g., distributing fake alerts).

2.2.1. External Attacks. External attacks have their origin outside the monitored network and can be classified as CIDS *disclosure* and *evasion* attacks. Disclosure attacks aim to detect the presence of CIDS monitoring points in the network as preparation for subsequent evasion attacks to bypass the CIDS.

In Shinoda et al. [2005], a method for the discovery of passive monitors that publish their results publicly via the Internet is presented. The assumption is that these results (e.g., periodically updated graphs that visualize the top targeted ports) provide enough information to trace the location of monitors. A similar but more active approach from Shmatikov et al. also assumes public CIDS output and introduces the concept of *probe-response attacks* [Shmatikov and Wang 2007]. Hence, the attacker carries out specifically adapted attacks so that the produced alerts of the CIDS contain a unique mark. These marks are then used to identify monitoring points. In Bethencourt et al.

[2005], passive sensor detection algorithms are presented that are based on probe-response attacks.

Furthermore, Rajab et al. [2006] describe techniques for live population sampling and methods for building sophisticated malware. In this case, the malware would try to spread intelligently over the Internet by first targeting only active IP space addresses and also by avoiding disclosure by CIDSs. The first part is done on-the-fly via sampling techniques in the different IP layers, accompanied by sending messages (e.g., ICMP packets). For avoiding disclosure, the assumption is that there is an increased probability that malware also attacks passive CIDS monitors while continuously choosing random IP addresses for further propagation. To prevent this, Rajab et al. propose the creation of malware with knowledge about the IP address ranges used by passive IDS monitors. Thus, such malware utilizes offline information regarding monitors that has been acquired with probe-response attacks. In addition, the authors also discuss the idea of malware that actively uses probe-response attacks during its infection phase.

Although CIDS disclosure techniques are interesting, they assume some kind of feedback path from the CIDS to the attacker (e.g., attack results are visible on a public website). Without such a feedback path, there are no automated techniques for an adversary to successfully disclose monitoring points of a CIDSs.

After disclosing the presence of a CIDS, attackers can try to evade the local detection mechanism of the monitoring points that can be either signature- or anomaly-based. Signature-based evasion techniques modify attacks so that they no longer match the known signatures (Cheng et al. [2011]). Anomaly-based evasion techniques (e.g., Debar et al. [1999], Wagner and Soto [2002], Tan et al. [2002], and Fogla et al. [2006]) try to masquerade attacks as legitimate and thus normal behavior. Because evasion attacks focus on bypassing the IDS detection mechanism, they can be universally applied to IDSs and thus are not restricted to CIDSs. For this reason, they are beyond the scope of this article.

2.2.2. Internal Attacks. Internal attacks refer to malicious behavior originated from within the monitored network. An insider can be classified as either a *malicious monitor* that is part of the CIDS monitoring topology (monitoring level) or a *malicious host* inside the monitored network.

Once a host is compromised within the protected network, a covert channel can be set up between this host and an external entity. A covert channel tries to hide the very existence of any communication [Lampson 1973; Zander et al. 2007] by using a channel that is usually not intended for communication (e.g., timing information in between packets or unused bits in the IP header). Covert channels presume a compromised host within the protected network, and, depending on the specific channel used, they can theoretically evade any CIDS.

It gets even worse when the attacker has compromised a component of the CIDS (e.g., a CIDS monitor) because this enables the adversary to launch subsequent attacks: When the attacker has successfully infiltrated a CIDS, it can disclose other monitors. Furthermore, a malicious user can compromise additional CIDS components or exploit vulnerabilities in the CIDS protocol to let its own compromised components take over more important positions or functions in the CIDS overlay, for example by producing fake alerts, accusing other monitors of being compromised, or by conducting supporting DoS attacks on other monitors. In addition, multiple compromised monitors can collude to take down or compromise the rest of the monitored network [Fung 2011]. Moreover, on the basis of a compromised monitor, the attacker can easily bypass the CIDS and thus reduce its accuracy by selectively forwarding alerts to other monitors (e.g., by suppressing alerts for specific detected attacks).

A generic countermeasure against malicious insiders in distributed systems is the adoption of reputation systems [Resnick et al. 2000; Marti and Garcia-Molina 2006], such as EigenTrust [Kamvar et al. 2003]. In this case, each of the monitors establishes a certain trust level based on its detection behavior or other defined properties. Whenever the trust level drops below a specified threshold, the monitor is considered nontrustworthy and specific measures can take place (e.g., blacklisting of the specific monitor).

Specifically adapted to CIDSs, Fung et al. [2009] propose a trust management model that is based on Bayesian probabilities. In more detail, when monitoring points distribute their alert data, they also send request messages to determine the trustworthiness of other monitors. This is achieved by a probabilistic model whose purpose is to measure the satisfaction level of the received response messages. A number of similar trust mechanisms for CIDSs to cope with insider attacks have been proposed [Fung et al. 2008; Duma et al. 2006; Sen et al. 2008; Gil Pérez et al. 2013].

Although the use of such mechanisms protects the system from many of the aforementioned attacks, it creates new opportunities for the attacker. A well-known problem with reputation systems is the exploitation of the system itself when a highly trustworthy peer(s) is compromised, which is called a *betrayal attack* [Fung et al. 2008]. If the trust value of the compromised peer is not quickly degraded, the overall accuracy of the system will be affected. In another variant, a so-called *sleeper attack* [Brinkmeier et al. 2009], a malicious peer first behaves benignly over time to establish a certain reputation level before it carries out the actual attack.

The topology of a CIDS might be completely static and preconfigured or, when monitors are added to the system dynamically, a strict access control for them can be enforced. However, in the highly unlikely case of a CIDS that is open and allows the dynamic inclusion of additional monitors, an attacker can launch a *sybil attack* [Douceur 2002] by adding a multitude of malicious monitors to the system [Fung 2011]. These can be used to establish a more detailed view of the CIDS topology and to prepare subsequent attacks, for example, to degrade the detection accuracy of the CIDS, to out-vote honest nodes, to perform whitewashing of malicious peers, and to compromise additional monitors. However, such attacks could be considered rare because most CIDS topologies are usually rather static, and CIDSs may enforce strict authentication mechanisms for new peers.

2.2.3. Attack Summary. In this section, we described external and internal attacks on CIDSs. External attacks can disclose the presence of an IDS and decrease its detection accuracy by evasion attacks. Alternatively, an attacker can issue DoS attacks to degrade the service availability of the CIDS or to keep it busy with spoofed alerts to camouflage its own attacks. An attacker who successfully compromises a host in the protected network can launch a multitude of additional attacks, such as setting up a covert channel to evade the IDS and thus decreasing its detection accuracy (e.g., to hide an export of sensitive data). It gets worse when the attacker compromises a CIDS monitor because this allows it to bypass the whole IDS, degrade its detection accuracy, or, in the worst case, to bring it down completely.

Internal attacks are even more effective when combined with external ones. Thus, compromised hosts or monitors may provide information to the outside that is used to prepare subsequent external attacks. For instance, a sophisticated attack may include an evasion technique to compromise a peer inside the monitored network without triggering any alerts. Afterward, the adversary could use covert channels [Zander et al. 2007] to send sensitive data outside the protected network.

Table I summarizes the aforementioned attacks with respect to the requirements given in Section 2.1. Any kind of attack on an CIDS is also an attack on its main

Table I. Relationship between Attacks on CIDSs and Requirements

CIDS Attacks	Accuracy	Overhead	Scalability	Resilience	Privacy	Self-config.	Interoper.
Signature Evasion	✓	✗	✗	✓	✗	✗	✗
Anomaly Evasion	✓	✗	✗	✓	✗	✗	✗
Denial of Service	✓	✓	✓	✓	✗	[✓]	[✓]
Insiders (Compromised Monitor)	✓	✓	✗	✓	✓	✗	✗
Insiders (Compromised Host)	✓	✓	✗	✓	[✓]	✗	✗
Covert Channels	✓	✗	✗	✗	[✓]	✗	✗

Note: Check marks ✓ indicate a relation between an individual requirement and IDS attacks, whereas checkmark symbols in brackets [✓] show an indirect relation. Finally, ✗ shows the absence of any relationship.

task—namely, the detection of attacks on the protected system—and thus an attempt to degrade the *Accuracy* of the respective CIDS. *Interoperability* and *Self-Configuration* are functional requirements and hence do not directly relate to the aforementioned attacks. However, DoS attacks may affect them. *Overhead* issues could arise through malicious monitors, DoS attacks, or even network-based insider attacks. Furthermore, *Resilience* is related to most CIDS attacks, whereas *Scalability* is affected only by DoS attacks, in the context of the system not being able to support additional monitors during such an attack. Attacks on the *Privacy* of CIDSs and user data are mostly related to malicious monitoring points because the exchanged CIDS alert data may contain sensitive information. For instance, this could be the case when several organizations use a single and interconnected CIDS. However, insiders and covert channels may also indirectly affect the *Privacy* of the involved participants in the monitored network because they can disclose sensitive data to unauthorized, external parties.

3. DESIGN SPACE AND BUILDING BLOCKS OF CIDSs

An efficient and holistic CIDS that fulfills all aforementioned requirements needs to be designed carefully. In the process, a multitude of challenges arise (e.g., minimizing the exchanged data by maximizing the detection accuracy, deciding which monitors should exchange information, and identifying the most efficient membership management architecture for the monitors).

To structure the solution space for such a system, we propose a disjunction of the CIDS into five main building blocks, as shown in Figure 3. We discuss each building block, its design space, and the arising challenges in the remainder of this section.

3.1. Local Monitoring

Local monitoring can take place on either the *host* or *network level*. On the host level, this requires monitoring of local activities to identify malicious behavior, which presumes monitoring functionality on all hosts in the network/system. In contrast, by monitoring at the *network level*, an entire network can be protected by deploying monitoring points only at strategically selected network locations (e.g., close to the ingress and egress routers). Combinations of host- and network-level monitoring are also feasible and will increase the amount of monitored data, thus allowing for more fine-grained attack detection in a CIDS.

Monitoring can be classified as either *passive* or *active*. Passive monitoring corresponds to scanning local activity or the locally observed network traffic. In active

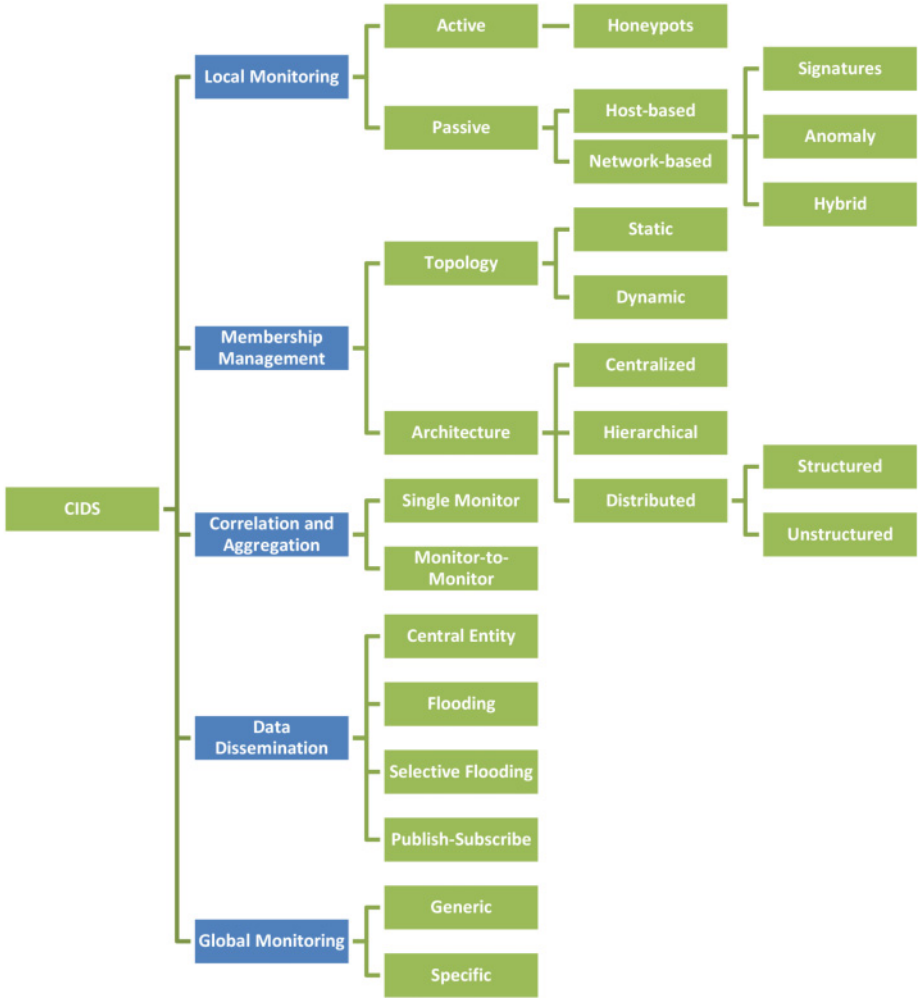

Fig. 3. Taxonomy of CIDSs.

monitoring, so-called *honeypots* are used to emulate the presence of vulnerable systems as promising attack targets. Because they have no productive use, any interaction with them can be classified as an attack. Hence, honeypots produce no false positives, although the false-negative rate can be high.

Detection engines are the individual mechanisms used in the analysis units of an IDS to detect attacks in the data collected by passive sensors. Such detection is either *signature-based* or *anomaly-based*; however, combinations of both mechanisms are also possible. For instance, Bro IDS [Paxson 1999] contains, beyond signature-based, modules that can be utilized for anomaly detection. Signature-based detection requires existing signatures for an attack and thus is unable to detect unknown attacks. Well-known examples of signature-based IDSs include Snort [Roesch 1999] and Suricata[1] [Albin and Rowe 2012]. Anomaly-based detection requires us to create a model of the

[1]http://suricata-ids.org.

normal behavior of the system. Each deviation from this model is then interpreted as an anomaly and thus as an attack. Hence, an anomaly-based detection can also detect unknown attacks. A comprehensive survey of anomaly-based detection methods is given in Chandola et al. [2009] and Garcia-Teodoro et al. [2009].

3.2. Membership Management

Membership management has the task of ensuring overall connectivity of the monitoring overlay of the CIDS by managing neighborhood relations between monitors. Depending on the CIDS, such membership management can result in static or more dynamic overlays that allow the dynamic inclusion and exclusion of monitors.

In the simplest case, the connections in the CIDS overlay are static and predetermined. Hence, an administrator needs to be involved whenever new components are added to the system. Alternatively, the CIDS overlay can be set up dynamically. This can be done either via a central server that has a global view of the system or via a membership management protocol that runs at each monitor and that operates on local knowledge only. This classification of *static* and *dynamic* overlays is also shown in Figure 3. Nevertheless, as can be seen in Sections 4–7, the majority of CIDSs adopts a dynamic overlay architecture.

Because the membership management controls the overlay neighborhood of CIDS components, it can also enforce a certain structure on it. Hence, the arising monitoring overlay can be either centralized, hierarchical, or completely distributed. In a *centralized CIDS*, all monitors are directly connected to a central analysis unit. A *hierarchical CIDS* arranges all monitors in a hierarchy that is rooted in a central analysis unit. Hence, monitors in lower tree levels report to the monitors above in higher tree levels. In addition, *hierarchical CIDSs* include approaches that make use of a number of supernodes. *Distributed CIDSs* prevent any SPoFs by deploying monitors in a flat overlay without exposed components like a central analysis unit.

Furthermore, the membership management can either have an *unstructured* overlay ID space or it can enforce a *structure* in case an additional location service is required; for example, on the basis of a Distributed Hash Table (DHT) [Androutsellis-Theotokis and Spinellis 2004]. Thus, a structured ID space would provide the advantage of a guaranteed broadcast and search functionality. However, for CIDSs, this requires us to map monitored data to a one-dimensional ID space. Therefore, multidimensional alert correlation (cf. Section 3.3) cannot be achieved easily by structured CIDS overlays. Depending on the observed attack scenario, selecting the right pattern as key for the DHT is crucial. For instance, a CIDS whose purpose is to be able to detect attacks originating from the same source has to use the source addresses from the monitored packets as the DHT key.

3.3. Correlation and Aggregation

Once data have been obtained and analyzed by the local monitoring, possible alerts need to be correlated and the monitored data need to be aggregated for later dissemination to other monitors or analysis units (cf. Section 3.4).

We distinguish between *single-monitor* and *monitor-to-monitor* correlation mechanisms. *Single-monitor* correlation correlates alerts/data locally at each monitor without sharing this information with other monitors. Hence, plain alerts or locally correlated alerts are shared either directly with an administrative interface of the CIDS or with a central analysis unit that carries out further correlation. *Monitor-to-monitor* correlation enforces sharing of alerts/data with other monitors that try to correlate this information with local information. Therefore, such a correlation technique requires sharing alerts or even more detailed data with other monitors. For this reason, sharing

blacklists of malicious IP addresses still remains a *single-monitor* correlation approach because this information is not input to the local correlation with local data.

Resulting and correlated alert patterns can have multiple dimensions, which is a major challenge for collaborative intrusion detection [Vincent Zhou et al. 2009]. For example, to detect an attack that is conducted from several source nodes simultaneously, it is not sufficient to correlate alerts solely based upon the IP addresses of the attack sources. Moreover, if more than one system is attacked at the same time, an alert correlation via the IP addresses of the victims is also not meaningful. Hence, a plethora of different patterns and combinations of them are imaginable for alert correlation. For instance, combinations of source IP, destination IP, protocol, source port, destination port, and even payloads of monitored packets can be used.

Alert correlation techniques in general can be classified into the following four different approaches [Zhou et al. 2010; Elshoush and Osman 2011]:

—*Similarity-based correlation* approaches (e.g., Valdes and Skinner [2001], Debar and Wespi [2001], and Cuppens [2001]) correlate alerts based upon the similarity of data or alert attributes. The similarity of two datasets is reflected by a score that is computed by similarity functions. Depending on the produced score, the data are then either correlated or not.

—*Attack scenario-based* approaches take causality into account when correlating data/alerts. Thus, they allow detecting complex attacks that take place in several steps. Such approaches (e.g., Dain and Cunningham [2001], Garcia et al. [2004], and Eckmann et al. [2002]) usually require one to establish an attack database. Furthermore, most of these approaches need to be initialized by a training dataset. Therefore, they provide high accuracy for known attacks, but fail in detecting unknown attacks.

—*Multistage alert correlation* techniques aim to detect unknown multistep attacks. Most such approaches presume the existence of relations among the different stages of an attack. Thus, they presume that an attack is conducted to prepare for another one. Multistage alert correlation requires building up a library of attack steps. Depending on the overall attack, multiple steps are then mapped/correlated to attack scenarios.

—*Filter-based* approaches (e.g., Porras et al. [2002]) attempt to filter irrelevant data or alerts to reduce the number of false positives in CIDSs. For that, alerts are prioritized according to their impact on the protected system. Thus, such approaches require a detailed description of the system to be protected (e.g., its network topology and the deployed operating systems) that is not always available. Moreover, the accuracy of the alert correlation depends on the level of detail provided in the system description.

3.4. Data Dissemination

Correlated alerts and aggregated data need to be efficiently distributed. The data to be disseminated can range from alerts to monitored data at all possible kinds of aggregation granularity. Especially CIDSs that focus on the detection of highly tailored and targeted attacks require data sharing beyond simple alert dissemination.

Data dissemination is heavily influenced by the CIDS architecture and thus by the applied membership management. Centralized CIDS have a predefined and directed flow of information, namely from monitors to the analysis unit. Decentralized CIDSs arrange their monitors in a hierarchy with a strict bottom-up flow of information. This hierarchy can be either completely static or changing dynamically (e.g., on the basis of the monitored data).

In contrast, distributed CIDSs provide a flat monitoring overlay and the highest level of freedom in exchanging data between monitors. Data dissemination in distributed CIDS can either result in flooding the entire CIDS overlay or in a *selective flooding*

by random walks [Vishnumurthy and Francis 2006] or gossiping approaches [Ganesh et al. 2003]. A selective, yet more intelligent flooding of a distributed CIDS overlay is provided by using *publish-subscribe* methods. Hence, monitors subscribe to other monitors for specific information (e.g., alert data). Thus, this establishes groups of monitors that are interested in the same kind of information. Within such groups, data can be exchanged in both directions and thus are not limited to the direction from publisher to subscribers. In distributed and DHT-based CIDSs, subscribers can send information to respective publishers (e.g., via a reverse multicast on top of DHT-based publish-subscribe).

3.5. Global Monitoring

Isolated IDSs fail in detecting highly distributed attacks because they cannot link distributed malicious events detected at different IDS instances. Therefore, they might not detect if malicious events are part of the same attack (e.g., a large-scale network port-scan). Therefore, from the perspective of an isolated IDS, one malicious port scan event may not seem to be worth reporting, and the attack and its severity might remain unnoticed.

For the detection of distributed attacks, a global monitoring mechanism is required and is built on collaboration and information exchange between monitors. Thus, this global monitoring is based on the data correlation and aggregation from Section 3.3 and, in the end, represents the detection capabilities of the respective CIDS. Depending on the specific scope of the CIDS, the global monitoring can vary from being generic (i.e., the system monitors and detects every possible attack), to specific (focused on malware spreading attacks, access control violations, etc.).

The remainder of this article contains a detailed survey of existing CIDS systems, starting with an overview of centralized CIDSs in the subsequent section.

4. CENTRALIZED CIDSs

In a centralized CIDS, monitors send all their information directly to a central analysis unit that either applies detection algorithms and/or alert correlation algorithms on the overall data. These systems are widely used because they provide high accuracy rates at low architectural complexity. However, in most centralized CIDSs, monitors are usually configured manually. Furthermore, such systems do not scale with the number of monitors and thus cannot protect large networks. In addition, because all data are collected and all analysis is done at one central unit, they might not be applicable for collaboration across different organizations due to privacy issues.

4.1. DIDS

Snapp et al. proposed Distributed Intrusion Detection System (DIDS) [Snapp et al. 1991] as one of the earliest centralized CIDSs in literature. DIDS tries to detect malicious activity over the monitored network and create an overall score of its security state. The DIDS architecture combines distributed monitoring with a centralized data analysis. DIDS consists of three basic components: the *DIDS director* that represents a central analysis unit, *host monitors*, and *network monitors*.

Network monitors observe all packets that are transmitted in their observed network segment. They apply simple host analysis techniques (e.g., checks of certain services such as *rlogin* and *telnet*) and utilization of heuristics to identify potentially intrusive behavior. The host monitor is responsible for the monitoring of a particular host. This unit also conducts a preliminary event analysis to decide which of the alerts should be forwarded to the director. The existence of a host monitor is not mandatory because

the network monitor can report network activities of hosts. In addition, at both monitor levels, aggregation is done by removing nonsignificant or OS-specific data before sending them to the central analysis unit. The main component of DIDS is the director, a centralized expert system that receives all alerts from host and network monitors. At this point, data are aggregated via a rule-based expert system, analyzed, and then the system decides whether there is a security breach on a certain host or a large-scale attack on the whole system. Finally, some correlation techniques are applied in DIDS. For instance, DIDS creates a unique ID for each monitor entering the monitoring environment. Afterward, any malicious activity related to this particular ID is consider part of the same attack.

DIDS applies only simplistic detection techniques that can probably be easily evaded by a sophisticated adversary. Hence, the accuracy can be rather poor. Moreover, another disadvantage of this system is its lack of self-configuration mechanisms. Furthermore, as the communication and computation overhead at the director increases with an increasing size of the monitored network, DIDS does not scale. The director component in DIDS is a SPoF and thus violates the *Resilience* requirement.

4.2. SURFcert IDS

SURFcert IDS[2] is a centralized IDS based solely on honeypots. *SURFcert's IDS* main scope is to create a large-scale CIDS that exhibits zero false positives. The system comprises multiple monitoring points, so-called *passive sensors*, that forward all their traffic via pre-established Virtual Private Network (VPN) tunnels to a centralized analysis unit, the so-called *tunnel/honeypot* server. At the tunnel server, the traffic is then analyzed by one or more honeypots, and the results are stored on a separate logging server.

In terms of accuracy, the exclusive use of honeypots results in a zero false-positive rate because any interaction with these systems is considered to be an attack. Nevertheless, honeypots cannot detect all attacks because they presume an interaction of the attacker with the honeypot. SURFcert IDS uses *Nepenthes* [Baecher et al. 2006], its successor *Dionaea,*[3] the *Kippo*[4] SSH honeypot, and *Argos* [Portokalidis et al. 2006] as a secure system emulator. All of these honeypots can only emulate certain ports and protocols. Hence, they cannot cover all possible attacks. This also illustrates why honeypots should not be used as the only detection method in an IDSs, but rather as an additional detection technology. Moreover, both computational and communication overhead of the tunnel server increase proportionally with the increasing number of sensors. Thus, the tunnel/honeypot server is not only a SPoF, but it also renders the system unscalable. In addition, there is an absence of alert correlation and aggregation mechanisms because all alert data is transferred to the central analysis unit. Finally, *SURFcert IDS* does not provide significant global monitoring capabilities because it is only able to present an overview of the local detected attacks (along with some statistics).

4.3. CRIM

Cooperative Intrusion Detection Framework (CRIM), introduced in Cuppens and Miège [2002], is a centralized cooperative module that obtains data from monitors or rather isolated IDSs. CRIM's scope is to analyze alerts and subsequently attempt to identify the adversaries' next possible steps. It provides functions for managing, clustering, merging, and correlating alerts and thus takes over the task of a central analysis unit

[2]http://ids.surfnet.nl.
[3]http://dionaea.carnivore.it.
[4]http://code.google.com/p/kippo.

in a CIDS. Monitors send their alerts in the standardized IDMEF data format [Debar et al. 2007] to the CRIM module.

To process the received data, an alert management function converts data to a set of tuples, which are then saved in a relational database. Afterward, an alert clustering function generates clusters of alerts based on a relation of similarity [Cuppens 2001]. The similarity relation between two alerts is created by an expert system and is based on the classification the alert, time, source, and target. Clusters are inserted into an alert merging function that creates new global alerts. The global alerts consist of the alert data collected from each cluster. Subsequently, global alerts feed a correlation function which conducts further analysis and creates a set of possible actions that might be performed by the adversary based on the current alert data. Finally, an intention recognition function is used to provide the administrator with attack information and the possible next steps of the attacker.

On the one hand, much of the authors' work is focused on correlation methods, which seriously reduces overall overhead. On the other hand, the multiple merging and correlation that is performed by several functions may lead to excessively abstract alerts and create a limited detection coverage and thus a poor accuracy. Furthermore, the proposed similarity-based correlation mechanism will not be able to relate sophisticated attacks. For instance, a slow distributed attack from different sources to different parts of the monitored network would remain undetected. Finally, the use of IDMEF as a standardized language for exchanging alerts is an advantage in terms of interoperability.

4.4. DIDMA

Distributed Intrusion Detection system using Mobile Agents (DIDMA) [Kannadiga and Zulkernine 2005] is a CIDS for the detection of distributed attacks on large networks. DIDMA makes use of static agents that act as local monitors. In addition, mobile agents exist that are responsible for alert dissemination as well as correlation and aggregation of data. Moreover, DIDMA contains a centralized entity that maintains lists of hosts experiencing similar attacks.

A DIDMA network can be seen as a static overlay in which local agents communicate with mobile agents. These local agents act as host monitors and generate alerts whenever malicious activity is detected, which also includes a classification of the type of the detected attack. Based on this classification, a global list of IP addresses is kept of nodes affected by the same type of attack (e.g., a DoS attack). Whenever an alert is generated by an agent, the IP address of the respective host is added to the list. Upon the occurrence of malicious activity and to detect a possible intrusion in the network, a central entity creates a mobile agent that can be transferred to other network positions. For each identified attack, the mobile agent updates the global list with IP addresses of other hosts with the same kind of suspicious activity. A mobile agent is then sent out and subsequently visits all hosts that are listed for the same type of attack. During this process, the mobile agent aggregates and correlates information from the visited hosts, updates the global list, and generates alerts when detecting any specific attack. In the end, alerts are sent to a central user interface for further analysis.

DIDMA utilizes a central entity for creating a global view of the overlay, as well as for dispatching mobile agents, and this entity represents a SPoF. DIDMA utilizes a signature-based detection algorithm and therefore cannot detect unknown attacks. Furthermore, the system requires a valid classification of the alerts for the agents to operate properly. Because this is not always possible, overall accuracy may be low. The system's overhead is highly affected by the number of hosts under attack, which corresponds to the number of hosts added to the global list. In addition, overhead rises when a high percentage of the detected attacks cannot be aggregated. In both of these

worst-case scenarios, DIDMA may produce considerably high communication overhead. DIDMA's resilience can be seriously affected in the case of malicious agents. Furthermore, if a host is compromised, an adversary could also compromise mobile agents.

4.5. Summary

Centralized IDSs usually provide high accuracy rates. Their main disadvantages are the lack of scalability in terms of the number of supported monitors and the SPoFs that are posed by the central analysis unit. Nevertheless, due to their higher accuracy, centralized IDSs are widely used in small to medium-sized corporate networks. DIDS [Snapp et al. 1991] was one of the first centralized approaches, and many systems followed its basic architecture. In addition, *CRIM* [Cuppens and Miège 2002] focuses on the correlation and aggregation of alert data. More recent approaches like *SURFcert* IDS provide interesting enhancements (e.g., honeypots) as an additional detection mechanism, whereas others, like DIDMA, make use of mobile agents.

5. DECENTRALIZED CIDSs

Decentralized CIDSs organize monitoring points or several different self-contained IDS deployments hierarchically in a tree that is rooted at a central analysis unit. Within the tree, preprocessing and correlation of monitored data takes place. On the basis of this correlated data, a distributed analysis for attacks takes place. However, when data aggregation in a hierarchy takes place, then information is lost at each level of the hierarchy. As a result, highly distributed and sophisticated attacks may remain undetected.

5.1. GrIDS

Graph Based Intrusion Detection System (GrIDS) is intended to protect large networks from actively propagating malware [Staniford-Chen et al. 1996; Cheung et al. 1999], but it can also detect attacks on individual hosts. The network is split into several zones, called *departments*, that are organized in a tree structure. Each department contains one analysis unit and several network and host monitors that perform intrusion detection and subsequently send their data to the analysis unit. Hence, each host in GrIDS belongs to a department, whereas the departments are controlled by parental departments, thus creating a hierarchy. Moreover, each department contains two special modules: a *software manager* and a *graph engine*. The software manager is responsible for the management of the local hierarchy status, as well as the monitors within a department. The overall tree hierarchy is ensured by a centralized hierarchy server. Finally, GrIDS provides the ability to make dynamic changes in the hierarchy via the utilization of a user interface.

The graph engine receives input from monitors within a department and thereupon establishes activity graphs that represent hosts and their network activities between each other. These activity graphs are first analyzed locally and afterward, they are aggregated and passed upward to the parental department and its graph engine. At this point, all information from child departments is merged, and graphs with coarser resolution are established. Suspicious behavior is detected on the basis of user-given detection rules that are expressed through a policy language.

The usage of detection rules suggests that, in terms of accuracy, the system would only detect attacks that are a violation of a security policy. Therefore, sophisticated or unknown attacks may still remain undetected. In addition, the aggregation mechanism may not be able to detect a widespread attack that progresses slowly because only attacks that occur within a short time period can be detected. Due to the division of the overall network into departments and their hierarchical organization, GrIDS is scalable in terms of protecting networks of arbitrary size. However, the hierarchy server that controls and maintains this hierarchy is an SPoF and serves as a potential

bottleneck. GrIDS is vulnerable to DoS and insider attacks, as are most of the centralized and decentralized CIDSs discussed in this survey. Finally, the system exhibits a built-in privacy protection mechanism due to the way it handles its hierarchy. In more detail, each department is only able to observe activity that is restricted within its boundaries.

5.2. AAFID

Autonomous Agents For Intrusion Detection (AAFID) is a hierarchical CIDS proposed in Balasubramaniyan et al. [1998] and Spafford and Zamboni [2000]. AAFID does not focus on the detection of specific types of attacks but rather acts as a framework in which different detection engines may be utilized. The system consists of *agents*, *transceivers*, and *monitors*. With respect to our terminology, agents act as monitors, while transceivers and monitors act as analysis units.

Agents are stationary and in the AAFID architecture, it is not foreseen that they migrate between different hosts. Each host can contain multiple agents that perform event monitoring and afterward send their reports to a transceiver. For instance, an agent could monitor for large numbers of port scans targeting a protected host. As soon as it detects this kind of activity, it will generate a report and send it to a transceiver. The authors claim that a variety of detection engines can be used in the agents (e.g., the IDIOT IDS [Crosbie et al. 1996]). *Transceivers* are entities that supervise all local agents, analyze their reports, aggregate the findings, and report them to one or more monitors. Moreover, transceivers have full control over the agents and can start, stop, and (re)configure them. *Monitors* can audit more than one transceiver. Because monitors receive alerts from all over the network, they can perform a data correlation over multiple hosts. However, the authors [Balasubramaniyan et al. 1998; Spafford and Zamboni 2000] do not give further details on that. Monitors can also be organized hierarchically, so that lower level monitors report to higher level monitors. Finally, a central monitor on top of the hierarchy communicates with a user interface.

AAFID utilizes a static hierarchical tree structure with a designated monitor taking over the root position and thus representing an SPoF. Hence, the system is not resilient against attacks or failures of these entities. Moreover, despite the fact that the authors regard low overhead and resilience against failures as important requirements, they do not address them in the implementation of their prototype. Finally, data dissemination as well as data correlation and aggregation are not addressed by them.

5.3. EMERALD

Event Monitoring Enabling Responses to Anomalous Disturbances (EMERALD) is another hierarchical CIDS [Porras and Neumann 1997]. It is designed for the monitoring of large enterprise networks and focuses on the detection of unauthorized access in domain resources. The system distinguishes three different layers: *service analysis*, *domain-wide analysis*, and *enterprise-wide analysis*. The service layer covers the detection of attacks across services and components within a single domain. The domain-wide analysis layer monitors multiple services and components. The enterprise-wide layer on top of the other layers, attempts to detect malicious activity across multiple domains.

Each of the aforementioned layers contains EMERALD monitors that use both signature-based and anomaly-based detection engines. Data dissemination in EMERALD is achieved via a subscription-based communication scheme. In more detail, each monitor may subscribe to others through a client/server-based asynchronous model and receive the respective alert data automatically. In addition, authors suggest that, to ensure the security of the messages exchanged between monitors, a public key authentication may be used. Finally, there is subsequent work on alert correlation

techniques [Valdes and Skinner 2001; Porras et al. 2002], as seen in Section 3, that have been tested in an EMERALD environment.

The hybrid detection engine used in EMERALD ensures a high accuracy for both known and unknown attacks. The system does not provide any mechanism for the detection of insider attacks. Regarding interoperability, EMERALD provides an API that can be used to interconnect different monitoring tools. Nevertheless, this requires additional effort by the user of the system. In addition, no standardized data format for information exchange with other IDS is used. Finally, according to the authors, ensuring reliable data delivery may increase the overall overhead of the subscription-based data dissemination mechanism.

5.4. HIDE

Hierarchical Intrusion DEtection (HIDE) is another approach described in Zhang et al. [2001], one that mainly focuses on applying novel anomaly detection techniques. *HIDE* arranges its monitors in a static hierarchical tier structure and employs anomaly detection via statistical preprocessing and neural network classification. Each tier in HIDE contains multiple monitors that are the so-called *Intrusion Detection Agents (IDAs)*. Each performs monitoring either on the host or network level.

An IDA collects network traffic or host events and abstracts them to statistical variables and reports. These reports are then statistically checked and compared against the reference model maintained in the IDA. Afterward, the result is taken and fed into neural network classifiers for further analysis and to determine if the traffic is normal or not. Finally, reports for higher tiers are generated, and information is displayed via a local user interface.

Because HIDE uses anomaly detection techniques, higher false-positive rates can be anticipated. In addition, the experimental results from Zhang et al. [2001] indicate that low-volume attacks are hard to detect. Moreover, the authors mainly concentrate on describing the detection algorithm and on the selection of the best neural network. For this reason, many details are missing to properly assess HIDE according to the requirements from Section 2.1.

5.5. Summary

Decentralized CIDSs are intended for the protection of large networks by overcoming the scalability problems of centralized CIDSs. However, most of the observed decentralized CIDSs accomplish this goal only partially because they usually contain one or more SPoFs. Moreover, decentralized CIDSs aggregate and correlate the data from lower levels and pass them over to the next level. At each level, the amount of data is reduced at the cost of lost information, which can result in lower accuracy compared to a centralized approach. Most of the proposals in this category focus either on novel architectures (GrIDS, AAFID) or detection algorithms (HIDE). In these terms, EMERALD, although an early approach, is the most complete solution with respect to our requirements.

6. DISTRIBUTED CIDSs

A distributed CIDSs architecture contains no central component or hierarchy because the tasks of the central analysis unit are distributed to all monitoring points. As a result, such a system, which follows the P2P design principle, can scale with any number of monitors and thus can protect large networks. Moreover, the lack of a strict hierarchy as in decentralized CIDSs provides more freedom in interconnecting monitors with each other and thus can be a benefit when encountering sophisticated and highly distributed attacks. However, depending on the specific distributed CIDS,

this can also be a drawback: Because there is no hierarchy, there is no component in the CIDS that has a global view of the protected network.

6.1. Structured CIDSs

Structured CIDSs impose a structure on the participating monitors by organizing them using a DHT. A DHT provides guaranteed broadcast and search functionality when storing data in a distributed manner. Most structured CIDSs described here make use of DHTs for the efficient storage of attack-related information (e.g., maintaining distributed blacklists). Others (e.g., INDRA) use a DHT for organizing their monitoring points.

However, this requires structure in terms of a fixed overlay neighborhood that is based upon the IDs of the CIDS monitors. Hence, this does not allow for flexible overlay connections. Moreover, since DHTs require that the data to be stored are mapped to the ID space of the DHT, any multidimensional data have to be reduced to single-dimension data. Therefore, when storing IDS data in a DHT, it is necessary to select a single property as a key for the DHT. Hence, only single-attribute lookups and no complex multi-attribute searches are feasible. Moreover, as a consequence of the strict ordering of nodes and data in the ID space, most DHT algorithms and implementations cannot provide strict locality properties. Due to this, privacy issues may arise when storing the monitored data.

6.1.1. INDRA. Intrusion Detection and Rapid Action (INDRA) is a P2P-based CIDS approach proposed by Janakiraman et al. [2003]. The system does not focus on any particular type of attacks, but rather on generic malicious activity detection. It organizes its monitors, so-called daemons, in a *Pastry*-DHT [Rowstron and Druschel 2001] and uses *Scribe* [Castro, Miguel, Druschel, Peter, Kermarrec, A.-M., Rowstron 2002] as publish-subscribe mechanisms for managing data sharing between daemons.

Monitors in INDRA act as both monitors and analysis units. When an attack is detected by an INDRA daemon, a proactive or reactive defense action takes place. An INDRA daemon consists of the different following subcomponents:

—*Watchers* detect suspicious activities on a host or network level.
—*Access controllers* are responsible for taking action against particular users (e.g., denying access to an account that is marked as compromised).
—*Listeners* aggregate the alerts generated by watchers and convey them to the access controllers.
—*Reporters* communicate with other hosts by means of sending and receiving alerts to and from other hosts, respectively.

The authors do not describe the detection mechanism used by *Watchers*. The information dissemination in INDRA is handled by *Scribe*. For every attack category, a *Scribe* group is created, and nodes can subscribe to these groups. For instance, nodes may subscribe to the *Scribe* groups for *SSH* and DoS attacks. In addition, the authors claim that alternative models, such as rumor-spreading, can be used for data dissemination, but without providing concrete suggestions on how to deploy them.

INDRA tries to improve its accuracy by allowing administrators to create plugins for new attacks. However, this manual intervention by the administrator needs significant effort because these plugins have to be written manually (as code). Furthermore, a compromised monitor can reduce the accuracy of INDRA by producing fake alerts as a form of a DoS attack. For example, consider the case when a compromised peer claims that another peer within the trusted network is compromised. One of the main suggested defensive mechanisms is the creation of a blacklist. Hence, all peers in the

network will insert a suspicious peer into their list, thus blocking it from any further communications.

6.1.2. LarSID. Zhou et al. describe Large Scale Intrusion Detection (LarSID), a P2P-based CIDS based on a publish-subscribe mechanism [Zhou et al. 2007, 2005]. Every peer in the system is a monitor (via the usage of local IDSs) and also an analysis unit that creates a list with suspicious IP addresses and distributes it to the P2P network.

In order to be able to share alert information between different peers, the system employs a publish/subscribe mechanism on top of a DHT; that is, a modified Pastry [Rowstron and Druschel 2001] DHT named *Bamboo* [Rhea et al. 2004, 2005]. The alert data in LarSID are in the form of lists of attackers' IP addresses. Each peer in the monitoring network is responsible for maintaining a watchlist for its local subnetwork, correlating subscription messages, and also generating notification messages regarding the identified malicious IP addresses. The system also utilizes a threshold policy. In more detail, if a certain number of monitors have flagged an IP address as malicious, then notifications are sent over the network. Otherwise (i.e., the number of detections of an IP is below the threshold), a new entry is created in the monitor's watchlist.

As a distributed CIDS, the system scales to large networks. This is also supported by the experimental results in Zhou et al. [2005]. However, as reported in Zhou and Leckie [2008], certain nodes can become overloaded when a large number of attacks is originated from the same IP address. LarSID assumes that all involved peers in the monitoring network are to be trusted. For this, it utilizes a Public Key Infrastructure (PKI) to ensure that participating nodes are authenticated. Moreover, communication between all monitors takes place over SSL. The main disadvantage of such an approach is its global monitoring capabilities. LarSID can only detect attacks that involve a common source or destination IP address.

Similar approaches. Many CIDS approaches have been proposed that are similar to the LarSID, such as *Komondor* [Czirkos and Hosszú 2012], *Wormshield* [Cai, Min, Kai Hwang, Yu-Kwong Kwok, Shanshan Song 2005], the P2P-based CIDS of Marchetti et al. [2009], and Cyber Disease DHT (CDDHT) [Li et al. 2006]. All of these proposals are similar in terms of their structured CIDS architecture, although the underlying DHT implementation may differ. Moreover, they exhibit differences with respect to their specific purpose as well as in their key selection for the DHT. In more detail, their global detection varies from worm detection and containment (Wormshield) and DoS attacks, port scans, worms, and botnets (CDDHT) to more flexible ones (Komondor [Marchetti et al. 2009]). Finally, RepCIDN [Gil Pérez et al. 2013] is another DHT-based CIDS that mainly focuses on the construction of a reputation mechanism (cf. Section 2.2.2) to handle internal attacks.

6.2. Unstructured CIDSs

Unstructured CIDSs provide great flexibility because no restrictions are imposed in selecting their overlay neighbors (peers to exchange data). Hence, this feature can be exploited in establishing flexible overlay relationships based on properties that are different from node IDs (e.g., based on similarities in the monitored data). However, because there is no structured ID space, as in structured CIDSs, data cannot be stored and retrieved efficiently as in structured CIDSs. For this reason, unstructured CIDS are not feasible for the distributed storing of attack-related information (e.g., blacklists) and efficiently looking them up again.

6.2.1. DOMINO. The Distributed Overlay for Monitoring Internet Outbreaks (DOMINO) is described in Yegneswaran et al. [2004]. It utilizes a hybrid architecture with three kinds of entities: *axis overlay*, *satellite communities,* and *terrestrial*

contributors. Axis nodes act as both monitors and analysis units. Satellite communities and terrestrial contributors act as additional monitoring points that send their alert results to axis nodes for further analysis.

Axis nodes are the central component of the system and are connected via an unspecified overlay network. The axis nodes are assumed to be especially trustworthy and use a PKI for mutual authentication. Moreover, to counter insider attacks and fake alerts, each of them can enforce a threshold filtering upon received data. Satellite communities are smaller networks of satellite nodes that locally implement a version of the DOMINO protocol. They are organized in a hierarchy that is always led by an axis node. Finally, terrestrial contributors expand the system by adding non-trustworthy peers who supply summaries of their detected attacks without implementing the DOMINO protocol.

DOMINO utilizes signature-based IDSs, firewall rules (for intrusion response), and also honeypots for intrusion detection. Active sinks are nodes that monitor a large number of unused IP addresses; they have a low false-positive rate that makes it possible to produce signatures for unknown attacks. Finally, the alert messages are represented in XML format and are exchanged periodically.

The hierarchical structure and the combined usage of both network-based IDSs and honeypots increase the overall accuracy of DOMINO. In addition, the alert messages are structured via XML, which ensures interoperability between different systems. Significant communication overhead may arise if the alert's broadcasting period is shortened, compared to the one implemented (i.e., hourly alert broadcasting). Resilience against certain insider attacks is achieved by a static axis node architecture. However, this inflexibility in the axis overlay renders the system vulnerable to attacks on axis nodes. Moreover, multiple compromised and cooperating axis nodes can pose a threat to the overall system, especially because DOMINO contains no explicit countermeasure against insiders (e.g., a reputation system). Nevertheless, internal DoS attacks (e.g., by sending large amounts of alerts from a compromised axis node) can be mitigated by a threshold filtering at each axis node.

6.2.2. Neighborhood-Watching. A P2P-based CIDS that uses mobile agents in a neighborhood-watch approach has been described [Ramachandran and Hart 2004]. Nodes are arranged in an unstructured P2P network in which different kinds of agents are exchanged among peers to check for possible attacks.

Peers watch out for their neighboring peers and store critical information on each of them (e.g., checksums of critical data and operating system files, as well as system binaries). If an anomaly is detected by a neighbor of an attacked peer, a voting process among all its neighbors takes place. When the majority agrees that intrusive behavior has been observed, all neighbors will try to protect themselves and at the same time notify and warn other peers in the network. To gather knowledge about neighbors in the P2P network, each peer sends different kinds of agents to its neighbors. These agents can perform a variety of tasks at the visited systems, such as establishing checksums of data files or looking for signatures of known viruses or worms. Moreover, agents are also used when voting about intrusive behavior in an overlay neighborhood.

In addition to a discussion of the system, the authors give no further evaluation of their system, nor any details on the methods used for the overlay establishment. At the same time, communication overhead issues may arise when a large number of agents is sent over the network to perform checks. The overall accuracy of the system could be low because information and data sharing is only done within a neighborhood level. Moreover, the authors do not provide enough insights for the detection mechanisms utilized. However, the voting process could have a positive effect in reducing the false-positive ratio. Furthermore, the voting process can also provide resilience against

insider attacks. Finally, sending agents to other systems to check crucial files may conflict with the privacy requirement.

6.2.3. NetBiotic. *NetBiotic* [Vlachos et al. 2004] is a distributed CIDS that is based on the *JXTA* P2P framework [Gong 2001]. The focus of *NetBiotic* is not on detecting specific attacks, but rather on the fast creation of a network of interested peers for alert information exchange. Hence, the goal is to provide basic protection to participating peers (e.g., by detecting rapidly propagating malware). With respect to our terminology, each *NetBiotic* peer is both a monitor as well as an analysis unit and hosts a *notifier* and a *handler* component.

At each peer, the *notifier* reads from log files written by security-related applications (e.g., by a local IDS). On that basis, the notifier detects attacks, creates statistics of attacks, and finally transmits these statistics to other peers. In particular, the *notifier* calculates and transmits the percentage by which the average number of detected attacks differs from the average hits detected in earlier time intervals. If this percentage is significantly higher than a preconfigured threshold, the peer is considered to be under attack.

The *handler* is responsible for receiving messages from other peers and, if need be, taking action, such as modifying the security settings of the end-user applications or introducing new firewall rules. A significant difference from most other IDS proposals is that *NetBiotic* takes defensive actions only when the number of attacks detected is higher than the average—when an epidemic outbreak occurs. Moreover, one of the main features of *NetBiotic* its ability to flexibly adapt the security policy, for example, when the rate of past attacks is higher than the current ones.

High accuracy rates are not the main focus of *NetBiotic*. As mentioned earlier, many attacks might remain undetected if there is no significant difference on the overall detection percentage. Interoperability is partially achieved because the system's architecture is supposed to be compatible with any IDSs that is able to record its alerts in a log file. However, certain dependencies exist, such as the need for the incorporation of a parser to extract data from the log files. In addition, the countermeasures can be OS-specific. Finally, the decrease of the security level when there is a low attack detection rate appears interesting. However, this feature may be exploited by an insider to lower the overall security of a network and to attack it subsequently.

6.2.4. Trust-Aware CIDS. Duma et al. propose a trust-aware P2P-based overlay [Duma et al. 2006] for CIDS based upon the JXTA framework [Gong 2001]. This CIDS specifically addresses insider threats through the utilization of a trust-aware correlation engine and a dynamically adjustable trust management scheme. With respect to our terminology, each peer in this system is both a monitor and an analysis unit.

The key component in each peer is the event manager that informs other peers of intrusion attempts and receives alerts from other peers. In addition, this unit provides filtering capabilities based on existing rules. The alert dissemination is done through selective flooding to known peers. All exchanged alert messages are in IDMEF format. To create trust among monitors, a list is maintained for acquaintance peers. Hence, every communication between monitors (e.g., the exchange of alerts) is evaluated, and a score is generated for each peer. The higher the trust level of a peer, the bigger is its impact on others. The acquaintance list is dynamically updated and includes only the best available candidates that, in addition, had to pass a probation period. However, the trust mechanism requires that each peer is able to determine whether an intrusion event was genuine or a false positive. This cannot be always ensured, and it highly depends on the accuracy of the employed local IDS. For the prototype implementation in Duma et al. [2006], *Snort* was used. Because this is a signature-based IDS, the probability that a detected intrusion is actually an attack is high.

The system is resilient to most of the insider threats, which includes Sybil and newcomer attacks. Nevertheless, some attacks are still feasible, such as sleeper attacks in which a highly trusted and long-term participating peer suddenly turns malicious. Such a peer would threaten the system as long as it remains in the acquaintance lists of other peers and can send fake alerts to others. The prototype system given in Duma et al. [2006] makes use of the *Snort* IDS that cannot detect unknown attacks and that can be easily evaded as described in Section 2.2. However, the system is interoperable because Snort comes with IDMEF support for exchanging alerts.

6.2.5. Worminator. In Locasto et al. [2004] and Locasto et al. [2005], the P2P-based IDS *Worminator* is introduced whose peers act as monitors, and each of them hosts a network-based IDS. *Worminator* exchanges compressed information via Bloom filters [Broder and Mitzenmacher 2004] with peers that are selected by a distributed correlation scheduling algorithm called *Whirlpool*.

Bloom filters are an efficient one-way data structure and are used to ensure privacy and compactness of the produced alerts. Upon a local alert, the corresponding information (e.g., source IP address and source port) is inserted into a Bloom filter. Hence, the Bloom filter represents a compressed list of suspicious hosts, a so-called *watchlist*. The watchlist is shared via the *Whirlpool* algorithm that creates dynamic neighborhood relationships in the overlay. Only neighbors exchange alert data via Bloom filters. However, the authors do not provide in-depth details on how this distributed scheduling algorithm works.

Bloom filters are probabilistic data structures. False-positive matches are possible especially with an increasing filling degree. However, there can be no false negatives: An element either has been included in the Bloom filter or not. Hence, when the number of included elements increases, innocent hosts that have not been explicitly included in the Bloom filter also could be identified as malicious. As a result, this affects the accuracy of *Worminator* beyond the actual detection mechanism. Nevertheless, the exchange of Bloom filters significantly decreases the signaling overhead compared to exchanging the uncompressed input data. Overhead is also reduced via dynamic neighborhood formation in *Whirlpool*, at least in comparison to a full mesh distribution scheme or a random selection distribution scheme. As in many CIDSs, insider attacks are not covered by *Worminator*; therefore, malicious monitors can generate fake alerts and accuse other monitors of being malicious. Privacy of sensitive data from the distributed alerts can be partially achieved by the utilization of Bloom filters since data is compressed and hashed. In addition, the system uses a fixed list of participants for the alert correlation, so that only legitimate users are granted with Bloom filter access. However, an insider could gain access to the Bloom filter and be able to launch subsequent attacks. Such an attack could be to delete entries from it or to insert IP addresses of innocent hosts to accuse them of malicious behavior. Nevertheless, this can be avoided by protecting the Bloom filters via cryptographic means. Finally, an internal attacker could also query specific IPs to check if they had been detected by the Bloom filter.

6.2.6. Quicksand. *Quicksand* is a CIDS that applies a distributed pattern detection mechanism for connecting distributed events manifested on a number of hosts [Krügel et al. 2002]. In a global monitoring level, *Quicksand* utilizes hybrid detection algorithms locally and afterward builds up signatures, so-called attack scenarios, that are used for connecting attacks that are distributed over the monitored network. To achieve this, each peer in *Quicksand* is a monitor that contains local IDSs with some prefiltering capabilities and an analysis unit, the so-called *event correlation unit*. Correlation units are also responsible for intrusion response (e.g., reconfiguring firewalls upon the local detection of an attack). Because *Quicksand* focuses more on distributed pattern

detection, it does not presume a specific detection engine. Monitors can locally apply signature-based as well as anomaly-based detection [Krugel et al. 2002] techniques.

The system makes use of a centralized unit that stores new attack signatures and updates the signature databases of monitors. To represent signatures of distributed attacks, the authors introduce the Attack Specification Language (ASL) for describing subsequent intrusion steps as blocks of patterns. This allows *Quicksand* to express complex relationships between distributed events on different hosts. Whenever an event on one host induces communication that leads to another event on another host, both events can be ordered and are set into relation in ASL. Afterward, each attack scenario described in ASL is transformed to a directed and acyclic pattern graph. Nodes in the graph correspond to the distributed events that take place at different hosts. Connections and relationships are represented as edges, whereby one node constitutes a particular event, and its successor node is the immediate successor of that event in the ASL. Once established, the centralized unit disseminates these pattern graphs to the monitors. Finally, at each monitor, the respective event correlation unit can check for possible intrusions. For that, it receives prefiltered streams of events from the local IDSs and executes a distributed misuse detection algorithm that detects the occurrence of attack patterns on the basis of the pattern graph. In their prototype, the authors combine a *Snort*-like signature-based detection with an anomaly-based detection mechanism from Krugel et al. [2002]. To ensure compatibility, the system applies IDMEF [Debar et al. 2007] for exchanging information between detection and correlation units, as well as between different monitors.

In terms of accuracy, while hybrid detection can be used locally by *Quicksand*, this is not the case for its global distributed pattern detection scheme. In this case, signatures (attack scenarios) have to be created, so false negatives (i.e., related attacks that failed to successfully be connected) exist. Moreover, the system allows only tree-shaped patterns to be created globally. However, this decision might not be always realistic because it excludes other kinds of patterns that might be more suitable. The system remains scalable because detection and correlation are performed locally first. Subsequently, only necessary information is exchanged between monitors, without the need for involving a central party in the detection process. Finally, Quicksand uses IDMEF for better interoperability, and the authors explicitly provide mechanisms for the integration of third-party detection engines and IDSs.

6.3. Summary

Distributed CIDSs were created to overcome the scalability limitations of centralized CIDSs, as well as the limitations of hierarchical approaches. In such an architecture, no entity has a global view of the network. Thus, the challenge is to provide accuracy rates close to that of a centralized CIDS while protecting larger networks in which centralized intrusion detection is no longer feasible.

On the one hand, structured distributed approaches provide efficient storing and lookup functionality. At this level, proposals such as LarSID offer a scalable CIDS solution with a fair, yet one-dimensional, global detection block. On the other hand, unstructured systems have the ability to couple nodes on the fly but with a less efficient way to store and retrieve alert data. In this direction, approaches such as DOMINO and the Trust-aware CIDS provide some promising and interesting properties [Yegneswaran et al. 2004; Duma et al. 2006].

7. COMPARISON OF CIDS APPROACHES

In this section, we give an overall comparison of the surveyed solutions of CIDSs by focusing on the individual proposed requirements given in Section 2.1. Table II provides an overview of all CIDS surveyed in this article according to their employed

Table II. CIDSs and Their Building Blocks

CIDS	Local Monitoring	Correlation & Aggregation	Membership Management	Data Dissemination	Global Monitoring
DIDS	Host-based	Single Monitor	Centralized	Central Entity	Generic
SURFcert	Honeypots	✘	Centralized	Central Entity	Generic
CRIM	Network-based	Single Monitor	Centralized	Central Entity	Attack Prediction
DIDMA	Network-based, Signatures	Monitor-to-Monitor, Similarity	Centralized, Static	Selective Flooding	Generic
GrIDS	Host-based and Network-based	Monitor-to-Monitor	Hierarchical	Flooding	Malware Spreading
AAFID	Host-based	Monitor-to-Monitor	Hierarchical	?	Generic
EMERALD	Network-based, Hybrid	Monitor-to-Monitor, Filter	Hierarchical	Publish-Subscribe	Access Control
HIDE	Network-based, Anomaly	Single Monitor	Hierarchical, Static	Flooding	Generic
INDRA	Network-based, Signatures	?	Distributed, Structured	Publish-Subscribe	Generic
LarSID	?	Monitor-to-Monitor, Similarity	Distributed, Structured	Publish-Subscribe	Malware Spreading, DDoS
DOMINO	Network-based and Honeypots	?	Distributed, Unstructured	?	Generic
Neighborhood Watching	Network-based, Anomaly	?	Distributed, Unstructured	Selective Flooding	Generic
NetBiotic	Host-based	✘	Distributed, Unstructured	Selective Flooding	Malware Spreading
Trust-aware CIDS	Network-based, Signatures	?	Distributed, Unstructured	Selective Flooding	Insider Attacks
Worminator	Network-based, Hybrid	Single Monitor	Distributed, Unstructured	Selective Flooding	Malware Spreading
Quicksand	Network-based, Hybrid	Monitor-to-Monitor	Distributed, Unstructured	Selective Flooding	Generic

Note: ✘ indicate that the respective building block is not available; a question mark (?) indicates unknown cases.

building blocks (cf. Section 3). Table III provides a summary of our findings by listing all discussed CIDSs and comparing them to the requirements from Section 2.

Accuracy. The foremost task of a CIDS is the detection of attacks; therefore, maximizing the *accuracy* of this detection is the most important requirement for CIDSs. With respect to the proposed building blocks of a CIDS, accuracy is mainly influenced by the employed local detection mechanisms (local monitoring) and the detection mechanisms that operate on shared data (global monitoring). Moreover, essential for the global monitoring are the employed data correlation and aggregation mechanisms. These mechanisms preprocess and merge subsets of the data obtained from different monitors as a basis for the detection mechanisms operating on them. It is expected that the class of centralized CIDS provides a higher detection accuracy than a decentralized or distributed CIDS. Centralized CIDSs and their employed detection mechanisms can operate on the full dataset, whereas decentralized CIDSs operate on aggregated data, and distributed CIDSs employ detection mechanisms that operate on subsets of the monitored data.

However, the level of achieved accuracy is determined by the specific detection mechanism employed by the respective CIDS. To make it worse, most of the surveyed CIDSs lack an evaluation of their accuracy in detecting attacks. A comparison of the surveyed

Table III. Collaborative Intrusion Detection Systems and Requirements

	CIDS	Accuracy	Overhead	Scalability	Resilience	Privacy	Self-config.	Interoper.
Centralized	DIDS	∅	×	×	×	?	×	?
	SURFcert	∅	×	×	×	×	×	×
	CRIM	∅	✓	×	×	?	?	✓
	DIDMA	∅	×	∅	?	×	∅	×
Decentralized	GrIDS	×	✓	∅	×	✓	?	×
	AAFID	?	?	✓	×	?	?	?
	EMERALD	✓	×	✓	?	?	?	∅
	HIDE	∅	?	✓	?	?	?	?
Distributed	INDRA	∅	?	✓	∅	×	×	×
	LarSID	?	?	✓	×	×	✓	×
	DOMINO	✓	?	?	×	×	?	∅
	Neighborhood-Watching	?	?	✓	✓	?	✓	×
	NetBiotic	×	?	✓	×	×	✓	×
	Trust-aware CIDS	∅	?	✓	✓	?	✓	✓
	Worminator	?	✓	✓	?	∅	✓	×
	Quicksand	∅	∅	✓	×	?	?	✓

Note: Checkmarks (✓) indicate the fulfillment of the individual requirement, ✗ indicates their nonfulfillment, ∅ indicates their partial match, and a question mark (?) indicates unknown cases.

CIDSs would require quantitatively evaluating all surveyed CIDSs in a comparable setting, which is simply not feasible. For this reason, it is difficult to state which approaches meet the accuracy requirements.

Because centralized CIDSs operate on the full dataset, we presume them to meet the accuracy requirements. Decentralized and distributed CIDSs are considered to have a lower accuracy and thus meet the requirements only partially. However, several of the discussed systems focus more on the architectural level than on the actual mechanisms for detecting attacks. For this reason, we consider these systems (e.g., NetBiotic and AAFID) to have a rather low accuracy and are thus assumed not to meet the accuracy requirements. Among the decentralized approaches, we only assume EMERALD to have sufficiently high accuracy because it utilizes a hybrid detection mechanism.

Among distributed CIDSs, we presume INDRA and NetBiotic to have low accuracy in detecting attacks. INDRA makes use of simplistic detection methods, and, regarding NetBiotic, accuracy is not in the main scope of the system, but rather the creation of a network of peers for fast information exchange. The only distributed CIDS that we assume to provide good accuracy is DOMINO. This approach exhibits hybrid local monitoring via a combination of honeypots, dynamic firewall rules, and network-based IDSs. However, in DOMINO, this comes at the expense of a significant computational and communication overhead.

Overhead. The communication overhead created by a CIDS is a direct result of the employed data sharing and dissemination mechanisms. Among the discussed CIDSs, a multitude of techniques is used for this purpose: reverse multicast, publish-subscribe methods, and flooding mechanisms. However, all of these techniques have advantages

and come with inherent drawbacks. For instance, while publish-subscribe algorithms seem to be promising for a deployment in CIDSs, it is not trivial to select the subscription criteria on which basis monitors subscribe to data. Flooding mechanisms come at high signaling overhead but cause monitored data to be available throughout a CIDS overlay and thus can result in increased accuracy. There is a tradeoff between the tolerated or caused signaling overhead and other requirements, such as resulting accuracy.

As can be seen in Table III, we assume that most of the discussed CIDSs cannot meet the requirement of minimal overhead. However, to compare them with each other and to assess their overhead, an extensive quantitative evaluation of all discussed systems would be required. Because this is not possible, we can only base our assessment on the architecture of the observed systems, their design choices, and their limitations. For instance, the CIDS Worminator tries to minimize communication overhead by using Bloom filters and by exchanging them between monitors via a certain scheduling algorithm. However, the data reduction by Bloom filters comes at the expense of decreased accuracy.

Scalability. Scalability is a fundamental requirement because CIDSs are intended to protect large networks. From the CIDS building blocks, the membership management is assumed to have the biggest influence on the scalability of CIDSs. Centralized approaches, as the name implies, utilize a centralized membership management and thus cannot scale to large networks. The central analysis unit represents a SPoF and a bottleneck. There are also several decentralized CIDSs that employ centralized membership management (e.g., GrIDS) and thus do not scale as well. Scalable, decentralized systems with interesting architectures are the AAFID, which creates a hierarchical tree structure, and EMERALD, which divides its monitored space in a multilayer fashion. The class of distributed systems is assumed by definition to be scalable. All distributed CIDSs summarized in this article seem to be free of a SPoF and bottlenecks. However, it is again difficult to assess the scalability of the discussed CIDSs individually because most have not been described in sufficient depth.

Resilience. A CIDS has to be resilient to failures and attacks, which also includes insider attacks from compromised and malicious system components. Hence, when under attack, resilient CIDSs need to stay available or at least provide graceful degradation. By their nature, centralized CIDSs cannot be assumed to be resilient. An attack on their central component will bring down the system immediately. Decentralized CIDSs are more resilient against failure and attacks. However, a failure of the central analysis unit on top of their hierarchy can result in service unavailability. Although there are mechanisms for automatic restoration of tree structures in case of failures, none of the surveyed systems makes use of them. Furthermore, in many cases (e.g., HIDE), the overall architecture appears to be static. Most of the observed distributed CIDSs utilize well-studied P2P protocols for membership management (e.g., DHTs). These approaches are well studied and provide mechanisms for graceful degradation and fast restoration after failures. Hence, we assume them to be resilient to failures and external attacks.

However, most of the CIDSs discussed in this article are vulnerable to insider attacks. Only the *Trust-aware CIDS* [Duma et al. 2006] and the *Neighborhood-Watching* approach from Ramachandran and Hart [2004] provide countermeasures against malicious insiders. Thus, in addition to these, we assume no other of the surveyed systems to be fully resilient and thus we rate them only as partially resilient.

Self-configuration. Self-configuration is a prerequisite for a resilient system because it allows for automatic system restoration, such as after a failure or an attack has taken place. In addition, self-configuration avoids the manual and error-prone configuration

of CIDSs. Distributed approaches inherently provide self-configuration mechanisms, such as via well-researched P2P mechanisms. DIDMA is the only centralized CIDS that offers partial self-configuration methods (e.g., agents are able to operate and self-configure themselves in the case of a failure of their dispatcher). For all other centralized CIDSs and decentralized CIDSs discussed in this article, it is not clear to what extent they support self-configuration.

Privacy. Privacy in terms of not disclosing data to unauthorized sources is also an important prerequisite when deploying a CIDS in larger scale and across different domains. However, only a few of the discussed systems include privacy-protecting mechanisms. For instance, *Worminator* partially achieves this requirement by combining the use of Bloom filters along with a trusted list of participant peers. In addition, *GrIDS* exhibits a built-in privacy-protection mechanism due to the way it handles its hierarchical architecture. In more detail, each department is able to only observe activity restricted within its boundaries.

Interoperability. Finally, a CIDS should also be interoperable with other CIDS deployments. For that, several of the observed systems (e.g., the *Trust-aware CIDS* [Duma et al. 2006] or *CRIM* [Cuppens and Miège 2002]) make use of standardized formats for data exchange, such as IDMEF.

Summary. To sum up, centralized CIDSs have the potential to offer the highest accuracy, but do not scale. Hence, they can only be used to protect small infrastructures, such as small corporate networks. For larger networks, such as a smart grid or large corporate infrastructures, more scalable solutions are required. Decentralized CIDSs seem to be suitable at first sight, but they are vulnerable to attacks and provide a lower accuracy than centralized systems. In terms of overhead, decentralized CIDSs' performance is highly dependable in regards to the utilized building blocks for correlation and aggregation. Most of the distributed IDSs are scalable to large networks and are resilient to attacks, but this comes at the expense of more overhead and accuracy degradation compared to centralized systems. Hence, especially in this category, a lot of challenges remain to be addressed by future research.

However, especially in the class of distributed CIDSs, many interesting systems have been proposed so far. For instance, DOMINO utilizes a hybrid architecture by combining a P2P-based core of especially trustworthy nodes with less trustworthy monitors that are organized in hierarchies. Nevertheless, as mentioned in the detailed analysis of DOMINO in Section 6.2, this might create inconsistencies with regard to our requirements, especially with respect to resilience against insider attacks. In addition, approaches from the structured CIDSs class, such as LarSID, demonstrate the efficiency and scalability of P2P CIDSs when the detection of certain attacks is required (e.g., DDoS).

The main finding of this article is that none of the observed systems can satisfy all requirements for CIDSs. While most distributed CIDSs are scalable and resilient, we assume that distributed approaches provide a lower detection accuracy than centralized or decentralized CIDS and induce too much overhead for a practical deployment. However, we also admit that our analysis of the surveyed CIDSs has been difficult because almost none has been evaluated in large-scale or real-world environments. This clearly shows the need for more research in collaborative intrusion detection, especially in the area of distributed CIDSs.

8. CONCLUSION

In this article, we summarized the state of the art in collaborative intrusion detection. Here, we give a detailed summary of our article and briefly discuss the remaining research gaps that we identified in the area of collaborative intrusion detection.

8.1. Summary

This article gives an overview on the current state of the art in the area of distributed and collaborative intrusion detection. For that, we derive requirements for CIDSs when deployed in large-scale environments and discuss attacks that try to evade them. Moreover, we disassemble CIDS into five basic building blocks and discuss each of these blocks extensively. The result of this discussion is a taxonomy for CIDSs along which we summarize and discuss popular CIDSs from the state of the art with respect to the derived requirements and the identified attacks. We mainly classify CIDSs by their resulting communication architecture (i.e., the utilized membership management), which can be either centralized, decentralized, or distributed. For each class, we provide a further refined taxonomy and discuss representative approaches in detail.

Along with this classification, we provide a detailed qualitative discussion of CIDSs, which reveals that none of the surveyed systems fulfills all of our requirements for CIDS. Many CIDSs, especially the decentralized and distributed ones, have either been developed with a specific attack scenario in mind or provide little more than an exchange of alerts and simplistic alert correlation techniques on top of it.

Each class of CIDSs has its advantages and disadvantages. Centralized IDSs are a solid solution for the protection of small networks. Because they carry out intrusion detection on the complete dataset, they provide better detection accuracy than decentralized and distributed approaches. However, their signaling overhead increases proportionally with network size and the number of monitors and thus limits their scalability.

An interesting centralized CIDS is *SURFcert IDS*, which utilizes honeypots for intrusion detection. However, because honeypots can only detect attacks that directly interact with them, they should never be the only detection mechanism.

Contrary to centralized CIDSs, decentralized and distributed CIDSs are scalable to large networks because they can support an arbitrarily large number of monitors. Nonetheless, because they carry out intrusion detection on subsets of the data only, a lower accuracy than centralized CIDSs can be expected. In these architectures, there is no central analysis unit, as there is in centralized CIDSs, that has the complete view on the alert data. While decentralized systems lose information in each correlation/aggregation step toward the central analysis unit on top of the hierarchy, distributed systems completely lack any component that has a global view of the network. Hence, their detection accuracy is heavily influenced by their data dissemination, data aggregation, and data correlation methods. As a result, we consider most current decentralized and distributed CIDSs as not suitable for the detection of sophisticated and highly distributed attacks. Furthermore, all surveyed decentralized CIDSs still contain SPoFs and bottlenecks that render them vulnerable to attacks. Distributed CIDSs do not have this drawback but induce higher signaling overhead than the other two classes.

Interesting decentralized proposals include GrIDS and EMERALD. GrIDS introduces a novel architecture and detection mechanism based on the construction of attack graphs. EMERALD, one of the earliest proposals, provides a hybrid detection engine combined with a strict separation of the monitored subnetworks, which makes it applicable for a real-world deployment. Among the distributed CIDSs described in this article, Quicksand and DOMINO are the most promising approaches. Quicksand implements a distributed pattern detection mechanism for linking events from different host and therefore is able to detect complex distributed attacks. In addition, DOMINO, a network-based CIDS that uses a P2P overlay for communication among its components and combines conventional detection methods with honeypots, is another interesting example from this class.

8.2. Research Gaps and Future Work

As one major outcome of this survey, we identified a research gap in the area of CIDSs. There is no scalable and viable solution for carrying out distributed intrusion detection in large networks. So far, only centralized CIDSs seem to have practical relevance because some of them are already deployed for the protection of small to medium-sized networks, such as *SURFcert IDS* (cf. Section 4.2). However, they do not scale with the number of monitors, and their central analysis unit is a SPoF. Decentralized and distributed CIDSs allow scaling to large networks, but are still in early stages of development. Most are either restricted to specific attack scenarios only or employ simplistic methods for information exchange and data correlation. As a result, they cannot provide the same detection accuracy as centralized CIDSs. To overcome this drawback, additional research into data correlation techniques is required. The problem is that intrusion detection attempts to find anomalies in data without knowing exactly what these anomalies look like. Hence, data correlation techniques are required that do not presume knowledge about the attributes on which the correlation is performed.

In addition to better data correlation techniques, more sophisticated techniques for anomaly detection are required that operate on exchanged data. For example, such methods could establish common anomaly detection models in cooperation with different monitors.

Furthermore, especially distributed CIDSs can greatly benefit from ongoing research into distributed P2P algorithms. Most distributed CIDSs employ a DHT for storing data and performing alert correlation, such as that based on publish-subscribe techniques like *LarSID* [Zhou et al. 2007] (cf. Section 6.1.2). More recent P2P approaches like *Skipnet* [Harvey et al. 2003] provide much better locality properties than DHTs. Locality in the resulting overlay eases the coupling and information exchange of close-by monitors, thus decreasing signaling overhead. Furthermore, locality can also be linked with the privacy requirement (cf. Section 2.1), making it possible to deploy CIDSs even in highly diverse network environments in which the sharing of alert data across multiple domains is required.

In this article, we discussed CIDSs approaches only qualitatively. Due to the lack of available implementations of CIDSs, no extensive quantitative evaluation (e.g., regarding their provided detection accuracy) can be provided. To make it worse, there is no up-to-date dataset available that would provide a fair comparison. Most security researchers still rely on the DARPA intrusion detection dataset [Lippmann et al. 2000] from 1999, which is outdated regarding its traffic patterns and also with respect to the recorded attacks. Hence, additional research efforts also are required to obtain datasets that allow a fair evaluation of different CIDSs.

REFERENCES

Eugene Albin and Neil C. Rowe. 2012. A realistic experimental comparison of the suricata and snort intrusion-detection systems. In *Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 122–127.

Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)* 36, 4 (2004), 335–371.

Stefan Axelsson. 2000. *Intrusion Detection Systems: A Survey and Taxonomy*. Technical Report. Department of Computer Engineering, Chalmers University.

Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. 2006. The nepenthes platform: An efficient approach to collect malware. *Lecture Notes in Computer Science* 4219 (2006), 165–184.

Jai Sundar Balasubramaniyan, Jose Omar Garcia-fernandez, David Isacoff, Eugene Spafford, and Diego Zamboni Ý. 1998. An architecture for intrusion detection using autonomous agents. In *Proceedings of the IEEE Computer Security Applications Conference*. 13–24.

Bazara I. A. Barry and H. Anthony Chan. 2010. Intrusion detection systems. In *Handbook of Information and Communication Security*. Springer Berlin, 193–205.

John Bethencourt, J. Franklin, and M. Vernon. 2005. Mapping internet sensors with probe response attacks. In *Proceedings of the 14th USENIX Security Symposium*. 193–208.

Michael Brinkmeier, Mathias Fischer, Sascha Grau, and Guenter Schaefer. 2009. Towards the design of unexploitable construction mechanisms for multiple-tree based P2P streaming systems. In *Kommunikation in Verteilten Systemen (KiVS)*. Springer, Berlin, 193–204.

Andrei Broder and Michael Mitzenmacher. 2004. Network applications of bloom filters: A survey. *Internet Mathematics* 1, 4 (Jan. 2004), 485–509.

Rainer Bye, Seyit Ahmet Campete, and Sahin Albayrak. 2010. Collaborative intrusion detection framework : Characteristics, adversarial opportunities and countermeasures. In *Proceedings of the Workshop on Collaborative Methods for Security and Privacy (CollSec)*. 1–12.

Yu Chen Cai, Min, Kai Hwang, Yu-Kwong Kwok, and Shanshan Song. 2005. Collaborative internet worm containment. *IEEE Security and Privacy Magazine* 3, 3 (May 2005), 25–33.

Antony I. T. Castro, Miguel, Druschel, Peter Kermarrec, and A.-M. Rowstron. 2002. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications* 20, 8 (Oct. 2002), 1489–1499.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *Computer Surveys* 41, 3 (July 2009), 1–58.

Tsung-huan Cheng, Y. Lin, Yuan-cheng Lai, and Po-ching Lin. 2011. Evasion techniques: Sneaking through your intrusion detection/prevention systems. *IEEE Communications Surveys & Tutorials* 99 (2011), 1–10.

Steven Cheung, Rick Crawford, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, Je Rowe, Stuart Staniford-chen, Raymond Yip, and Dan Zerkle. 1999. *The Design of GrIDS: A Graph-Based Intrusion Detection System*. Technical Report. University of California at Davis.

Mark Crosbie, B. Dole, T. Ellis, Ivan Krsul, and E. H. Spafford. 1996. *Idiot-Users Guide*. Technical Report.

Frédéric Cuppens. 2001. Managing alerts in a multi-intrusion detection environment. In *Annual Computer Security Applications*. IEEE, 22–31.

Frédéric Cuppens and Alexandre Miège. 2002. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'02)*. IEEE, 202–215.

Zoltán Czirkos and Gábor Hosszú. 2012. Enhancing collaborative intrusion detection methods using a kademlia overlay network. In *Proceedings of hte 18th EUNICE/IFIP WG 6.2, 6.6 International Conference*, Vol. 7479. Springer, 52–63.

Oliver Dain and Robert K. Cunningham. 2001. Fusing a heterogeneous alert stream into scenarios. In *Proceedings of the ACM Workshop on Data Mining for Security Applications*. 1–13.

Herve Debar, David A. Curry, and Benjamin S. Feinstein. 2007. The Intrusion Detection Message Exchange Format (IDMEF). The Internet Engineering Task Force (IETF).

Hervé Debar, Marc Dacier, and Andreas Wespi. 1999. Towards a taxonomy of intrusion-detection systems. *Computer Networks* 31, 8 (April 1999), 805–822.

Herve Debar and Andreas Wespi. 2001. Aggregation and correlation of intrusion-detection alerts. In *Recent Advances in Intrusion Detection*. Springer, 85–103.

John R. Douceur. 2002. The sybil attack. In *Peer-to-Peer Systems*. Springer, Berlin, 251–260.

Claudiu Duma, Martin Karresand, Nahid Shahmehri, and Germano Caronni. 2006. A trust-aware, P2P-based overlay for intrusion detection. In *Proceedings of the International Conference on Database and Expert Systems Applications (DEXA'06)*. IEEE, 692–697.

Steven T. Eckmann, Giovanni Vigna, and Richard A. Kemmerer. 2002. STATL: An attack language for state-based intrusion detection. *Journal of Computer Security* 10, 1–2 (2002), 71–103.

Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman. 2011. Alert correlation in collaborative intelligent intrusion detection systems—A survey. *Applied Soft Computing* 11, 7 (Oct. 2011), 4349–4365.

Prahlad Fogla, Monirul I. Sharif, Roberto Perdisci, Oleg M. Kolesnikov, and Wenke Lee. 2006. Polymorphic blending attacks. In *Proceedings of the USENIX Security Symposium*. 241–256.

Carol Fung. 2011. Collaborative intrusion detection networks and insider attacks. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 2, 1 (2011), 63–74.

Carol Fung, Olga Baysal, Jie Zhang, Issam Aib, and Raouf Boutaba. 2008. Trust management for host-based collaborative intrusion detection. *Managing Large-Scale Service Deployment* 5273 (2008), 109–122.

Carol J. Fung, Jie Zhang, Issam Aib, and Raouf Boutaba. 2009. Robust and scalable trust management for collaborative intrusion detection. In *Proceedings of the International Symposium on Integrated Network Management*. IEEE, 33–40.

Ayalvadi J. Ganesh, A.-M. Kermarrec, and Laurent Massoulié. 2003. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computing* 52, 2 (Feb. 2003), 139–149.

Joaquin Garcia, Fabien Autrel, Joan Borrell, Sergio Castillo, Frederic Cuppens, and Guillermo Navarro. 2004. Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation. In *Information and Communications Security*. Springer, 223–235.

Pedro Garcia-Teodoro, J. Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security* 28, 1–2 (Feb. 2009), 18–28.

Manuel Gil Pérez, Félix Gómez Mármol, Gregorio Martínez Pérez, and Antonio F. Skarmeta Gómez. 2013. RepCIDN: A reputation-based collaborative intrusion detection network to lessen the impact of malicious alarms. *Journal of Network and Systems Management* 21, 1 (March 2013), 128–167.

Li Gong. 2001. JXTA: A network programming environment. *IEEE Internet Computing* 5, 3 (2001), 88–95.

John R. Goodall, Wayne G. Lutters, and Anita Komlodi. 2004. I know my network: Collaboration and expertise in intrusion detection. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*. ACM, 342–345.

Nicholas J. A. Harvey, Michael B. Jones, Stefan Saroiu, Marvin Theimer, and Alec Wolman. 2003. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, Vol. 4. USENIX Association, Seattle, WA, 1–14.

Mark D. Hill. 1990. What is scalability? *ACM SIGARCH Computer Architecture News* 18, 4 (1990), 18–21.

Ramaprabhu Janakiraman, Marcel Waldvogel, and Qi Zhang. 2003. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)*. IEEE, 226–231.

Peyman Kabiri and Ali A. Ghorbani. 2005. Research on intrusion detection and response : A survey. *International Journal of Network Security* 1, 2 (2005), 84–102.

Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. 2003. The eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*. 640.

Pradeep Kannadiga and Mohammad Zulkernine. 2005. DIDMA: A distributed intrusion detection system using mobile agents. In *Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. IEEE, 238–245.

Christopher Krügel, Thomas Toth, and Clemens Kerer. 2002. Decentralized event correlation for intrusion detection. In *Proceedings of the International Conference on Information Security and Cryptology (ICISC'02)*, Vol. 2288. Springer, Berlin, 114–131.

Christopher Krugel, Thomas Toth, and Engin Kirda. 2002. Service specific anomaly detection for network intrusion detection. In *Proceedings of the ACM Symposium on Applied Computing (SAC'02)*. ACM, 201–208.

Butler W. Lampson. 1973. A note on the confinement problem. *Communications of the ACM* 16, 10 (Oct. 1973), 613–615.

Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava. 2005. Intrusion detection: A survey. In *Managing Cyber Threats*. Vol. 5. Springer, 19–78.

Zhichun Li, Yan Chen, and Aaron Beach. 2006. Towards scalable and robust distributed intrusion alert fusion with good load balancing. In *Proceedings of the SIGCOMM Workshop on Large-Scale Attack Defense (LSAD'06)*. ACM, New York, 115–122.

Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. 2000. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks* 34, 4 (Oct. 2000), 579–595.

Michael E. Locasto, Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo. 2005. Towards collaborative security and P2P intrusion detection. In *Proceedings of the IEEE Workshop on Information Assurance and Security*. IEEE, 333–339.

Michael E. Locasto, Janak J. Parekh, Salvatore Stolfo, and Vishal Misra. 2004. *Collaborative Distributed Intrusion Detection*. Technical Report. Columbia University.

Mirco Marchetti, Michele Messori, and Michele Colajanni. 2009. Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale. *Lecture Notes in Computer Science* 5735 (2009), 475–490.

Sergio Marti and Hector Garcia-Molina. 2006. Taxonomy of trust: Categorizing P2P reputation systems. *Computer Networks* 50, 4 (March 2006), 472–484.

Vern Paxson. 1999. Bro: A system for detecting network intruders in real-time. *Computer Networks* 31, 23–24 (Dec. 1999), 2435–2463.

Phillip A. Porras, Martin W. Fong, and Alfonso Valdes. 2002. A mission-impact-based approach to INFOSEC alarm correlation. In *Proceedings of the Conference on Recent Advances in Intrusion Detection (RAID'02)*. Springer, 95–114.

Phillip A. Porras and Peter G. Neumann. 1997. EMERALD: Event monitoring enabling response to anomalous live disturbances. In *Proceedings of the National Information Systems Security Conference (NISSC'97)*. 353–365.

Georgios Portokalidis, Asia Slowinska, and Herbert Bos. 2006. Argos: An emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. *ACM SIGOPS Operating Systems Review* 40, 4 (2006), 15–27.

Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis. 2006. Fast and evasive attacks: Highlighting the challenges ahead. In *Recent Advances in Intrusion Detection*, Vol. 4219. Springer, Berlin, 206–225.

Geetha Ramachandran and Delbert Hart. 2004. A P2P intrusion detection system based on mobile agents. In *Proceedings of the Southeast Regional Conference ACM-SE*. ACM, 185–190.

Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. 2000. Reputation systems. *Communications of the ACM* 43, 12 (2000), 45–48.

Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. 2004. Handling churn in a DHT. In *Proceedings of the USENIX Annual Techincal Conference*. 127–140.

Sean Rhea, Brighten Godfrey, and Brad Karp. 2005. OpenDHT: A public DHT service and its uses. *ACM SIGCOMM Computer Communication Review* 35, 4 (2005), 73–84.

Martin Roesch. 1999. Snort-lightweight intrusion detection for networks. In *Proceedings of the USENIX Conference on System Administration*. 229–238.

Antony Rowstron and Peter Druschel. 2001. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Middleware 2001* (2001), 329–350.

Poly Sen, Nabendu Chaki, and Rituparna Chaki. 2008. HIDS: Honesty-rate based collaborative intrusion detection system for mobile ad-hoc networks. In *Proceedings of the 7th Computer Information Systems and Industrial Management Applications*. IEEE, 121–126.

Yoichi Shinoda, K. Ikai, and M. Itoh. 2005. Vulnerabilities of passive internet threat monitors. In *Proceedings of the 14th USENIX Security Symposium*. 209–224.

Vitaly Shmatikov and Ming-Hsiu Wang. 2007. Security against probe-response attacks in collaborative intrusion detection. In *Proceedings of the Workshop on Large Scale Attack Defense (LSAD'07)*. ACM, New York, USA, 129–136.

Steven Snapp, James Brentano, Gihan Dias, Terrance Goan, Todd Heberlein, Che-Lin Ho, Karl Levitt, Biswanath Mukherjee, Stephen Smaha, Tim Grance, Daniel Teal, and Doug Mansur. 1991. DIDS (Distributed intrusion detection system): Motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Security Conference*. 167–176.

Aditya K. Sood and Richard J. Enbody. 2013. Targeted cyber attacks: A superset of advanced persistent threats. *IEEE Security & Privacy* 11, 1 (2013), 54–61.

Eugene H. Spafford and Diego Zamboni. 2000. Intrusion detection using autonomous agents. *Computer Networks* 34, 4 (2000), 547–570.

Lance Spitzner. 2003. Honeypots: Catching the insider threat. In *Proceedings of the Computer Security Applications Conference*. IEEE, 170–179.

A. Srivastava, B. B. Gupta, A. Tyagi, Anupama Sharma, and Anupama Mishra. 2011. A recent survey on DDoS attacks and defense mechanisms. In *Advances in Parallel Distributed Computing*. Springer, 570–580.

Staniford-Chen, Steven Cheung Stuart, Richard Crawford, Mark Dilger, Jeremy Frank, James Hoagland, Karl Levitt, Christopher Wee, Raymond Yip, and Dan Zerkle. 1996. GrIDS—A graph based intrusion detection system for large networks. In *Proceedings of the National Information Systems Security Conference*. 361–370.

Kymie M. C. Tan, Kevin S. Killourhy, and Roy A. Maxion. 2002. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection*, Vol. 2516. Springer, Berlin, 54–73.

Alfonso Valdes and Keith Skinner. 2001. Probabilistic alert correlation. In *Recent Advances in Intrusion Detection*. Springer, 54–68.

Emmanouil Vasilomanolakis, Mathias Fischer, Max Mühlhäuser, Peter Ebinger, Panayotis Kikiras, and Sebastian Schmerl. 2013. Collaborative intrusion detection in smart energy grids. In *Proceedings of the International Symposium for ICS & SCADA Cyber Security*. Electronic Workshops in Computing (eWiC), 97–100.

Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. 2009. Decentralized multi-dimensional alert correlation for collaborative intrusion detection. *Journal of Network and Computer Applications* 32, 5 (Sept. 2009), 1106–1123.

Vivek Vishnumurthy and Paul Francis. 2006. On heterogeneous overlay construction and random node selection in unstructured P2P networks. In *Proceedings of the International Conference on Computer Communications (INFOCOMM'06)*. IEEE, 1–12.

Vasileios Vlachos, Stephanos Androutsellis-Theotokis, and Diomidis Spinellis. 2004. Security applications of peer-to-peer networks. *Computer Networks* 45, 2 (2004), 195–205.

David Wagner and Paolo Soto. 2002. Mimicry attacks on host-based intrusion detection systems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS'02)*. ACM, New York, USA, 255–264.

Vinod Yegneswaran, Paul Barford, and Somesh Jha. 2004. Global intrusion detection in the domino overlay system. In *Network and Distributed System Security (NDSS)*.

Sebastian Zander, Grenville J. Armitage, and Philip Branch. 2007. A survey of covert channels and countermeasures in computer network protocols. *IEEE Communications Surveys* 9 (2007), 44–57.

Zheng Zhang, Jun Li, C. N. Manikopoulos, Jay Jorgenson, and Jose Ucles. 2001. HIDE: A hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In *Proceedings of the IEEE Workshop on Information Assurance and Security*. IEEE, 85–90.

Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. 2005. A peer-to-peer collaborative intrusion detection system. In *Proceedings of the International Conference on Networks*. IEEE, 118–123.

Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. 2007. Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 80–89.

Chenfeng Vincent Zhou and Christopher Leckie. 2008. Relieving hot spots in collaborative intrusion detection systems during worm outbreaks. In *Proceedings of the 2008 IEEE Network Operations and Management Symposium (NOMS'08)*. IEEE, 49–56.

Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. 2010. A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security* 29, 1 (Feb. 2010), 124–140.

Received February 2014; revised November 2014; accepted January 2015