Collaborative Security: A Survey and Taxonomy

GUOZHU MENG, YANG LIU, and JIE ZHANG, Nanyang Technological University, Singapore
ALEXANDER POKLUDA and RAOUF BOUTABA, University of Waterloo, Canada

Security is oftentimes centrally managed. An alternative trend of using collaboration in order to improve security has gained momentum over the past few years. Collaborative security is an abstract concept that applies to a wide variety of systems and has been used to solve security issues inherent in distributed environments. Thus far, collaboration has been used in many domains such as intrusion detection, spam filtering, botnet resistance, and vulnerability detection. In this survey, we focus on different mechanisms of collaboration and defense in collaborative security. We systematically investigate numerous use cases of collaborative security by covering six types of security systems. Aspects of these systems are thoroughly studied, including their technologies, standards, frameworks, strengths and weaknesses. We then present a comprehensive study with respect to their analysis target, timeliness of analysis, architecture, network infrastructure, initiative, shared information and interoperability. We highlight five important topics in collaborative security, and identify challenges and possible directions for future research. Our work contributes the following to the existing research on collaborative security with the goal of helping to make collaborative security systems more resilient and efficient. This study (1) clarifies the scope of collaborative security, (2) identifies the essential components of collaborative security, (3) analyzes the multiple mechanisms of collaborative security, and (4) identifies challenges in the design of collaborative security.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Invasive software*; H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces—*Collaborative computing*

General Terms: Design, Performance, Security

Additional Key Words and Phrases: Collaborative security, taxonomy, privacy, trust, intrusion detection, spam, malware, information sharing

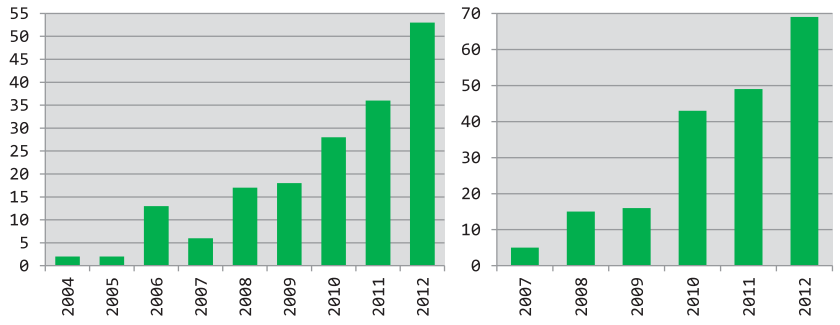
ACM Reference Format:
Guozhu Meng, Yang Liu, Jie Zhang, Alexander Pokluda, and Raouf Boutaba. 2015. Collaborative security: A survey and taxonomy. ACM Comput. Surv. 48, 1, Article 1 (July 2015), 42 pages.
DOI: http://dx.doi.org/10.1145/2785733

1. INTRODUCTION

In the last several years, cybersecurity attacks have increased the risk of property loss, privacy leakage, and a general disruption of daily life. Targeted attacks are consistently increasing year after year, with increases of 42% and 81% over the last 2 years,

This research is supported (in part) by the National Research Foundation, Prime Minister's Office, Singapore under its National Cybersecurity R & D Program (Award No. NRF2014NCR-NCR001-30) and administered by the National Cybersecurity R & D Directorate. This research is also partially supported by "Formal Verification on Cloud" project under Grant No: M4081155.020. This work is also partially supported by the A*STAR SERC grant (1224104047) awarded to Dr. Jie Zhang.
Authors' addresses: G. Meng; email: gzmeng@ntu.edu.sg; Y. Liu; email: yangliu@ntu.edu.sg; J. Zhang; email: zhangj@ntu.edu.sg; A. Pokluda; email: apokluda@uwaterloo.ca; R. Boutaba; email: rboutaba@uwaterloo.ca.

DOI: http://dx.doi.org/10.1145/2785733



(a) Published Items in Each Year (b) Citation in Each Year

Fig. 1. Trends of collaborative security based on Web of Science.

respectively [Symantec 2012, 2013]. Individual security once dominated the security area, but individual security systems must base all of their decisions and actions to prevent and react to attacks, and detect security vulnerabilities, on limited knowledge. Increasingly open and scalable networks, sophisticated attack techniques, and more frequent communication within distributed systems make it more difficult to provide an effective security service based on individual systems. Several significant threats to current security mechanisms and strategies include:

—*Slow reaction to new attacks.* Hackers are now exploiting zero-day vulnerabilities in order to silently download and install malware on the computers of victims. An example of this is the Java zero-day attack [Constantin 2013], which is aimed at Java 7 and may have started on January 2, 2013. There is not yet a complete patch from Oracle.
—*Inconspicuousness of distributed attacks.* Distributed attacks tend to remain hidden until they have made significant damage. According to recent news from *ChinaNews* [ChinaNews 2013], Android users are at risk of being infected by the largest botnet discovered to date, through which compromised smart phones can be manipulated to divulge confidential information, receive obnoxious advertisements, or launch distributed attacks.
—*Deficiencies in mobile environments.* Comparing to the conventional computing environment, there are many special features in mobile environments [Oberheide and Jahanian 2010]. Mobile devices lack centralized management and are in constant contact with the outside world, which makes them susceptible to attacks. Furthermore, the limited resources of mobile devices prevent them from adopting complex, comprehensive algorithms and technologies to prevent and detect attacks.

To cope with these challenges, researchers and vendors have proposed collaborative security [Seigneur and Slagell 2009], a new kind of security that coordinates nodes to perform specific security actions in order to enhance the security of networks or a whole system. Over the past few years, collaborative security has proven to be an effective and durable approach to detect vulnerabilities, prevent attacks, and protect sensitive information. More recently, research on collaborative security has markedly increased. As outlined in Figure 1,[1] the research related to collaborative security is attracting more attention, as is evident by the steady increase of research published in recent

[1]This data was collected by searching "collaborative botnet OR collaborative intrusion detection OR collaborative* malware OR collaborative* inside* attack* OR collaborative spam" in the Web of Science on March 14, 2013.

years. It is, and will continue to be, a hot topic in the security field for the foreseeable future. Collaborative security is constantly developing and continues to be applied to new security domains.

The success of collaborative security relies on not only its ability to address the challenges of traditional security but also the accuracy and efficiency of security analysis. An implementation of collaborative security must be mindful not to introduce new security vulnerabilities. For instance, communication channels could be susceptible to attacks, privacy may be divulged during collaboration, and the system itself could be subverted by an internal attacker. As an emerging concept, collaborative security is often misunderstood; the techniques and mechanisms in collaborative security are numerous, and the field lacks a state-of-the-art survey and comprehensive taxonomy. It is, therefore, significant and urgent to have a thorough and comprehensive study on collaborative security to help structure further research in this increasingly important area. To the best of our knowledge, this survey is the first to systematically analyze collaborative security from a computer science perspective. This article will focus on the scope of collaborative security, the fundamental components, mechanisms and techniques employed, interesting phenomena, and critical concerns when designing a collaborative security system. We will then clarify where and how to use collaborative security, and provide constructive solutions to specific issues. We will focus on:

—The scope of collaborative security, where we will propose a collaborative security framework based on the fundamental components of 44 investigated collaborative security systems. This also contributes to understanding the applications of collaborative security, and provides a basis for comparing different collaborative security systems.
—The domains of intrusion detection, spam filtering, malware blocking, the detection of internal attackers, and the detection of botnets. These systems are explained within this survey following the proposed framework. We will also consider additional information, such as enabling technologies and their merits, which help to provide a more complete view of collaborative security.
—The seven classifications in the developed taxonomy, which include analysis target, timeliness of analysis, architecture, network infrastructure, initiative, shared information, and interoperability. Within this taxonomy, we identify limitations, technologies and trends that can guide the implementation of collaborative security systems to guarantee aspects such as trust, privacy and scalability.
—Five challenges (including privacy, accuracy, scalability, robustness and incentive) in designing and developing collaborative security systems; we also analyze how these challenges can be addressed by further research.

The remainder of this article is organized as follows: Section 2 summarizes previous investigations of collaborative security. Section 3 presents a fundamental framework for collaborative security, which is comprised of the most common components discovered in our investigation. Section 4 investigates the threats emerging in collaborative security systems. Section 5 surveys and classifies different collaborative systems based on their security goals. Section 6 describes collaboration mechanisms from different aspects and creates a comprehensive taxonomy of collaborative security. Section 7 provides a discussion in which we talk about common phenomena, statistic features, critical difficulties, and development trends. Section 8 identifies challenges in collaborative security. Finally, Section 9 discusses possible areas for future research.

2. RELATED WORK

Although there have been earlier attempts to explore the paradigm of collaborative security and review associated methods, the scopes of such attempts are often restricted

to specific domains, which lack systematic analysis and classification. Collaborative Computer Security and Trust Management [Seigneur and Slagell 2009] is a collection of collaborative security-related research; however, the discussions therein lack detailed and insightful analysis and summarization.

Common building blocks of collaborative intrusion detection systems are identified in Bye et al. [2010] and include communication scheme, group formation, organizational structure, information sharing, and system security. The paper also discusses privacy preservation during sharing security-related information. In contrast to Bye et al.'s research, our article covers a considerably larger number of challenging issues and suggests promising solutions for them.

Two main challenges in designing a collaborative intrusion detection system are proposed in Zhou et al. [2010]. Their research surveys many coordinated attacks that traditional intrusion detection systems cannot detect. Zhou et al. introduce a new kind of intrusion detection system through a collaborative lens. Our work concentrates on attacks that collaborative security systems are best able to prevent and discusses how this collaboration can better solve these problems.

There are early works looking into specific aspects of collaborative security, however they do not consider the entirety of the topic. Chandola et al. [2009] survey multiple categories of collective anomalies and present key challenges for each category. They also investigate a series of methods to handle these collective anomalies as well as a thorough comparison between these methods. Elshoush et al. [2011], for example, discuss one particular field in collaborative intrusion detection systems: alert correlation. Their research surveyed a considerable number of applied approaches of alert correlation and presented the strengths and weaknesses respectively. Caruana and Li [2012] also conducted a survey of spam filtering approaches, specifically those dealing with collaboration, and provided a summary of the practical applications.

This article enhances previous research by providing a broader investigation of technologies. We propose a general framework of collaborative security, including intrusion detection, anti-spam, anti-malware, and botnet detection. In addition, we aim to caution researchers with potential problems within the collaborative security framework.

3. ANALYTICAL FRAMEWORK FOR COLLABORATIVE SECURITY

In this section, the scope of collaborative security is discussed, specifying the definition, the objective, and involved domains of collaborative security. We then discuss common components that have been identified as fundamental to collaborative security systems.

3.1. The Scope of Collaborative Security

In Seigneur and Slagell [2009], collaborative security is briefly defined as "Instead of centrally managed security policies, nodes may use specific knowledge (both local and acquired from other nodes) to make security-related decisions." As stated therein, the final objective of using nodes is to make security-related decisions. These decisions must happen in a community in which nodes can contribute their efforts to make the decisions more effectively and reasonably. Nodes should collaborate with each other, sharing some security evidence or analysis results, even (local) security-related decisions. Collaborative security is, therefore, a joint effort between multiple security systems through the sharing of security-related information to make more effective and reasonable decisions.

Collaborative security has been widely applied in many security domains, for example, intrusion detection, anti-spam, anti-malware, identification of insider attackers and detection of botnet. The application of collaborative security ranges from the desktop environment to the mobile environment, however, with the prerequisite skill of communication. Nodes in one community need to connect and communicate with each

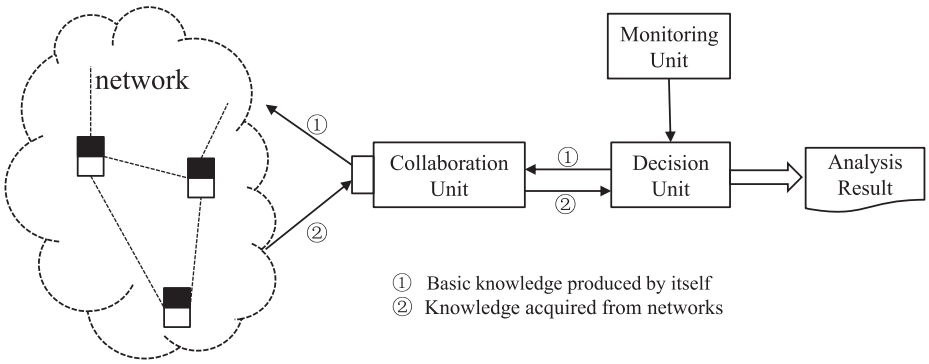


Fig. 2. The general architecture of collaborative security.

other as a precondition to perform a specific security-related task. Additionally, security is commonly regarded as evidence- and experience-based; therefore, more abundant information and advanced technologies are prone to better security-related decisions. This makes collaborative security prevalent in detecting attacks and protecting computing environments.

3.2. Building Blocks of Collaborative Nodes

Due to their common purpose, nodes in collaborative security systems generally share a common structure. Within this article, we provide an analytical framework for collaborative security, which serves as an internal backbone for summarizing and analyzing previous research. With the analytical framework, we submit different classifications as well as their strengths and weaknesses existing in collaborative security systems (see Section 6), which could facilitate the design of an effective and robust collaborative security system.

In a typical collaborative security system, an intrusion or attack violating pre-defined rules and restrictions can be captured by specific monitoring nodes. The attack information will subsequently be transferred to a unit with a more powerful analytical ability for confirmation. The information that cannot be handled will be disseminated to other security systems for collaborative analysis. For better communication, these systems should negotiate an agreement for exchanged data in advance. Four parts of a typical collaborative security system are shown in Figure 2. These are regarded as being fundamental and are described as follows:

—**Monitoring Unit:** This unit is the first inspection unit and producer of primary security-related data. As the activator of the whole process, the monitoring unit detects anomalies and potential threats based on preassigned rules or logics. The result will then be transferred to the successor collaboration unit. Normally, it can be deployed on either an endpoint host capturing the suspicious behaviors of local software, or an intermediate device analyzing abnormal network traffics.
—**Decision Unit:** This unit makes security-related decisions based on local observation and acquired knowledge from other nodes. It integrates algorithms and techniques to process the collected information, and eventually decides whether the captured anomalies are real attacks or not.
—**Collaboration Unit:** This unit is the core component in collaborative security systems. It shares local analysis results with other systems on the network (denoted as message ①). Similarly, the collaboration unit may also receive knowledge from the network (the knowledge is either the feedback on the enquired suspicions or

the possessed knowledge of attacks) denoted as message ② to facilitate the security detection. This unit should specify the communication mechanism and associated technologies among nodes.

—**Shared Information:** The shared information is a specific data structure, containing an abstract description of an operand or security evidence disseminated among nodes. Specifically, the shared information is always well-structured as being standard and commonly acknowledged by other nodes. In addition, constrained by the capability of the decision unit, the information may appear in many forms depending on how the decision unit processes it, which is discussed in Section 6.

Summary: The overarching goal of collaborative security is to make more effective and robust decisions. Compared to traditional security, collaboration units and shared information are unique. Therefore, the systems need to make extra communication efforts, normalizing the exchanged information. It is worth mentioning that for a robust collaborative security system (i.e. not suitable for all), there are always some mechanisms to prevent insider attacks. An example of this is trust management, about which we will provide a thorough discussion in Section 8.

4. SECURITY THREATS

In this section, we identify ten types of threats collaborative security aims to prevent. These threats are collected basically from two sources: (1) the surveyed literature in which certain threats are prevented by collaborative security systems and (2) the typical security threats from [MIT Corporation 2003a; Undercoffer et al. 2003; Igure and Williams 2008; Simmons et al. 2009; Microsoft 2013]. Some collaborative security systems aim to address the issues of general threats, such as intrusion and malware. We hence conclude the typical and concrete threats in terms of these common taxonomies of threats. For example, malware may cause the privacy leakage, or privilege escalation in an attack. Then, systems which can prevent malware can naturally resist the attacks of privacy leakage and privilege escalation. More details about the correlation can be found in Section 7.1. We have organized these threats in the following, based on the goal of the attacks.

Privacy Leakage. A potential risk of downloading online software is the possibility of exposing users' sensitive data such as account credentials, preferences, contacts, and so on. Attackers may use some techniques like brute-force attacks, man-in-the-middle attacks, and phishing tactics in order to steal sensitive data. Privacy leakage through downloading malicious software has been exacerbated in recent years on mobile devices with the rise in popularity of mobile applications. Sensitive information such as the device's identity, contacts, messages, and financial information are the main targets for hackers using malware programs. Specifically, sensitive information potentially attacked is twofold: (1) contacts, messages, personal information available on social networks; and (2) financial information can be directly accessed by malicious users. These are examples of explicit privacy, which is mentioned by many in the literature [Barkan et al. 2003; Schmidt et al. 2009; Enck et al. 2010; Reed et al. 2010; Schlegel et al. 2011; Arapinis et al. 2012; Grace et al. 2012]. Another kind of privacy noted is implicit privacy. Implicit privacy denotes the information that malicious users cannot directly use; in order for it to be beneficial, the attacker must analyze it in order to reveal valuable information. Using this kind of side-channel attacks, the hacker can extract secure information by analyzing video and audio data, timing, keystrokes, power consumption, and notifications of network connection. Kocher et al. [1999] find secret keys from tamper resistant devices from analyzing power consumption measurements. Schlegel et al. [2011] present an approach that can gather audio data from

on-board sensors and use it to recognize commercial credentials. Qian et al. [2012] use packet counter side channels to infer the sequential numbers used to launch inference attacks. Song et al. [2001], Vuagnoux and Pasini [2009], and Chen et al. [2010] refer to additional approaches.

Privilege Escalation. It is common to grant privileges to an application upon installment; however, vulnerabilities in these applications can result in an increase of privilege authorizations, data tampering or the disclosure of information. Permissions on Android, for example, must be explicitly identified and applications cannot access the device's resources until the installer grants it the required permissions. However, many malicious applications circumvent the permission mechanism and exploit indirect tactics to access sensitive resources. As Grace et al. [2012] describe, permission mechanisms can be infiltrated by malicious applications calling other applications that have their authorized permissions; RageAgainstTheCage, Exploid, and Zimperlich are three sorts of typical exploits of Android vulnerabilities that are employed to elevate the privilege of applications [Zhou and Jiang 2012; Jiang and Zhou 2013]. In addition, offline attackers can manipulate mobile devices into launching a distributed denial of service attack. We categorize these into threats of authorization violations, and such attack cases can be found in Dagon et al. [2004], Traynor et al. [2006], Cho et al. [2010], and Singh et al. [2010].

Authentication Violation. Authentication is a security scheme used to identify whether a user is as it claimed, using signature and encryption technologies. However, some malware may impersonate as other applications in order to carry out these particular behaviors. Examples of cases on authentication violation occuring in mobile devices can be found in Baltatzis et al. [2012], Fuchs et al. [2009], Qian et al. [2012], and Schmidt et al. [2009].

Spam. While it is sometimes treated more of an annoyance than a threat, by sending myriad messages (e.g., emails), attackers can post an advertisement or spread viruses through spam. From another prospective, they can result in high overhead of traffic, which can cause denial of service. Due to high profit and low technical requirements, spam has become one of most significant threats.

Routing Trap. Routing Traps occur when nodes claiming to transfer and forward packets fail to perform their duty, which will deny service to the associated nodes. Examples of this kind of attacks are: sinkhole attacks[2] [Krontiris et al. 2007b]; blackhole attacks[3] [Patcha and Mishra 2003], and; selective forwarding attacks[4] [Krontiris et al. 2007a]. These malicious nodes should be equipped with high-speed bandwidth, rapid reaction, and insidious tactics to entice normal nodes to consider them as the transmit point. However, the malicious nodes discard all or partial packets, leading to a black hole in the network in order to impede communication.

[2]The sinkhole attack occurs when a compromised node exploits the vulnerability of the routing algorithm, makes it as the relay node for as many nodes as possible. In the consequence, large portion of traffic will be forwarded to this node during the routing process. The attacker can subsequently launch more severe attacks, such as tampering and replaying.

[3]The blackhole attack is a compromised node playing the role of a relay. Each packet through this node will be withhold and cannot reach to its destination. This type of attack gives the impression of a black hole because the nodes it serves cannot get outside and communicate with other nodes.

[4]The selective forwarding attack is that a compromised node intentionally or randomly discards some packets to prevent their propagation in the network. Superior to the blackhole attack, selectively forwarding packets can avoid the awareness of its neighbours and reduce suspicion of its wrongdoing.

Denial of Service. In a denial of service attack, an attacker tries to make a host, or services on this host, unavailable to its intended users. Availability is a significant concern of service providers. An attacker may crash services on a host, for example, exploiting a vulnerability existing in a service to disturb its normal operation, and thereby making it unavailable to legitimate users; it can also flood a host by issuing a huge number of requests to prevent other users from connecting to the host. These kinds of attacks are exacerbated in mobile ad hoc networks because newly joined mobile devices, which may be potentially infected by Trojan horses, may form an uncontrolled botnet. Subsequently, they can launch a distributed denial of service attack to cause too high overload and eventual breakdown of the targeted host.

Deceptive Interaction. Attackers try to deceive innocent agents and convince them that they are communicating with a trusted principle. After obtaining the trust from these agents, attackers can launch further attacks. For example, one network node can spoof other nodes that it can redirect packages to the specific target in a routing process. However, it will definitely not accomplish the commitment as a relay node. Nodes, hence, cannot connect to the target as expected.

Malicious Code Execution. In this attack, malicious code is deployed somewhere in advance, and attackers can exploit existing vulnerabilities of systems to execute the malicious code. The malicious code is either a virus or a worm, which can further cause damage to the system [Kim et al. 2010].

Abuse of Functionality. To launch an attack, attackers may manipulate one or more functionalities of systems, which should not be used arbitrarily. By breaking this security policy, the attackers can alter or influence the normal behaviors of the system, or destroy the integrity of information. In short, this attack can be regarded that an attacker leverages the intended functionality to obtain the undesired outcome of the target system. For example, a rantankerous user may type incorrect passwords a specific number of times to lock out an innocent account [Microsoft 2014].

Resource Depletion. Every node in collaborative security systems has limited resources to perform its task, especially for mobile devices and sensors. It is even accentuated due to their limited computing power, storage and energy. Malware tries to occupy clock cycles of CPU, take up all storage or exhaust energy to affect other software's functionalities. Though there is a considerable development of physical hardware, computation power, memory capacity, and battery supply, it is still the bottleneck for mobile devices and sensor devices. The installed malware can exhaust the resources and affect the functionalities of other applications event cause the breakdown of the system. In Dagon et al. [2004], Nash et al. [2005], Racic et al. [2006], and Kim et al. [2008], battery life has been proven as a prominent shortcoming that the attackers likely use to make the device unavailable. Moreover, computation power [Pütz et al. 2001; Miettinen and Halonen 2006; Bye and Albayrak 2008; Becher 2009] and memory [Nash et al. 2005; Miettinen and Halonen 2006; Becher 2009] are both the enticing targets for attackers concluded from known attacks.

Summary: These types of threats provide a rich environment for collaborative security to emerge and develop. Unsurprisingly, the deficiencies and ineffectiveness of individual security dealing with these threats make collaborative security more attractive. Malware detection, for example, is usually based on malware signatures or anomalies (henceforth referred to as "knowledge") [Idika and Mathur 2007], from the prospective of methodology. Compared to the dramatically increasing number of malware variants, the increase of the knowledge occurring in an individual system is sluggish. Merging these security systems could assist in facilitating the timely prevention of the newest

kinds of malware. Moreover, some attacks (e.g., privacy leakage and privilege escalation) may bypass the protection of the security system through the collaboration of several attackers. A collaborative security approach has been found to be useful in detecting such attacks.

5. COLLABORATIVE SECURITY SYSTEMS

In this section, we present six types of collaborative security systems in terms of their security goals. We first summarize the existing research, as well as the improvement on previous works in a chronological order for each kind of collaborative security systems. We then provide a conclusive description based on the analytical framework, followed by a discussion on unique collaboration highlights and technologies.

5.1. Collaborative Intrusion Detection

Intrusion detection can help improve the security of networks and hosts by immediately reactions to attacks; this can then be divided into *H*ost-based Intrusion Detection (e.g., OSSEC [2013] and TripWire [2013]), Network-based Intrusion Detection (e.g., SNORT [2013] and Bro [2013]). However, individual power is not always enough. To enhance the effect of intrusion detection systems, sharing data, known as Collaborative Intrusion Detection Systems (CIDSs), is a good option. In this subsection, we investigate 12 collaborative security systems.

Indra, proposed by Janakiraman et al. [2003], is a typical CIDS with trusted nodes sharing security-related information. Each node equally contributes to the protection against intrusion attempts. The authors came up with three inventive 3-How problems in CIDSs: how to communicate with each other; how to trust shared information and senders, and; how to react to intrusions. These are the three underlying problems when designing and developing CIDSs. In the paper, Janakiraman et al. briefly introduced the measures taken to solve these problems.

Indra stresses the significance of information sharing, however, disregards the efficiency and the reasonability of communication. To foster the collaboration among intrusion detection systems and accelerate the look-up process, Yegneswaran et al. [2004] designed DOMINO (Distributed Overlay for Monitoring InterNet Outbreaks). The communication in DOMINO is guaranteed by employing a hierarchical architecture, in which the responsibilities vary from one node to another. Trusted axis nodes on the highest level are organized in a peer-to-peer manner; satellite nodes taking an axis node as the root form a hierarchical tree for the bottom-up message delivery; and terrestrial nodes, which are deployed at the bottom of the infrastructure, keep delivering the daily intrusion summaries to their superiors. Additionally, there is a certification authority distributing keys of cryptography that can ensure the trustworthiness of the messages. This design enables *DOMINO* to be secure, scalable, fault-tolerant, and facilitates data sharing.

Influenced by the biological immune system, Luther et al. [2007] propose a cooperative intrusion detection approach. The whole system is comprised of many individual artificial immune system (AIS) agents, which are organized in a novel manner called hybrid decentralized. By negative selection, each AIS agent chooses certain detectors during the training phase and exchanges detectors' status information, which can greatly improve the performance of detection as well as reduce false positives in anomaly detection.

In contrast to returning analysis results immediately as described earlier, a collaborative approach that collects security-related information afterward and proceeds with aggregation or correlation is called TRINETR. Yu et al. [2004] propose this collaborative architecture for multiple intrusion detection systems. It can collect alerts generated by IDSs by standardizing the intrusion alerts. In order to make the analysis more effective

and accurate, the system first determines the priority and affected systems of alerts by referring two bases: (1) network and host knowledge base (e.g., IP address and service ports) and (2) vulnerabilities knowledge base (e.g., CVE [MIT Corporation 2003b], Bugtrap [SecurityFocus 2003] and CERT [CMU 2004]). The system then finds the relationship among the alerts. By gathering, aggregating and correlating the alerts, the collaborative approach can find potential sophisticated attacks more macroscopically and precisely.

In addition, CIDSs are also widely used in Mobile Ad Hoc Networks (MANET) because they can significantly alleviate the limitation of resources. Zhang et al. [2003] proposed a collaborative technique for intrusion detection systems in mobile networks. Every node in mobile networks is deployed with an IDS. Any node that detects an intrusion or anomaly will confirm the attack with the collected evidence, and subsequently initiate a response. If it does not have strong evidence, it will initiate a global cooperative detection by sending state information of the intrusion to its neighbours. The state information represents the level confidence the node has about the likelihood of an attack. All the nodes together can then collaborate to decide if it is an intrusion of anomaly based on majority rule.

This approach does leave some issues, however; for example, anomaly detection will produce relatively high false alerts and nodes work in an inefficient way—all the nodes have to participate into the global intrusion detection process without any duty separation. Given this, Kachirski et al. [2003] proposed a distributed intrusion detection system for MANET based on mobile agent technology. Nodes are equipped with specific functions that are only responsible for some specific tasks, which can minimize the power consumption and processing time. In addition, clustering is also used in this system to reduce the workload of the network, whereby nodes are elected to monitor the network and make decisions accordingly. The segregation of duties can therefore maximize the utilization rate of nodes and minimize the consumption, thus making communication more efficient than employing the hybrid decentralized architecture (e.g., cluster).

Inspired by the work of Zhang et al. [2003], Albers et al. [2002, 2007] presented a distributed and collaborative architecture of IDS amongst mobile agents. The distribution of the intrusion mechanism was achieved by implementing a Local Intrusion Detection System (LIDS) on each node. Albers et al.'s work broadened the knowledge of the environment compared to [Zhang et al. 2003]. LIDSs share not only intrusion alerts, which are the detected intrusions on each local host, but also security data, the environmental information about the hosts. Moreover, the approach employs a trust-based mechanism to enhance the robustness of LIDS, where nodes behaving abnormally will be excluded from communities until they reauthenticate themselves.

To some extent, CIDSs are restricted by runtime resource constraints in MANET. To solve this problem, Huang and Lee [2003] proposed a cluster-based scheme (also mentioned in Albers et al. [2002], Kachirski and Guha [2003], and Anantvalee and Wu [2007]) for their CIDSs where periodically a node is elected as the intrusion detection agent for a cluster. It is claimed that most of MANET nodes are working uselessly unless the system is suffering intensives attacks. Therefore, to make it more efficient, the authors proposed cluster formation algorithms and a cluster-based intrusion detection scheme. The whole network can be divided into several clusters; one node is elected as the cluster head in each cluster, and then takes the responsibility for monitoring the whole cluster.

The organizing principle of clusters just mentioned is based on distance amongst mobile devices; however, there are other principles in forming a group in these IDSs. Bye and Albayrak [2008] presented a cooperation scheme named *Collaborative Intrusion and Malware Detection (CIMD)*: all nodes state their objectives and form into

groups in order to exchange security-related information in terms of these objectives. The authors gave a tree-oriented taxonomy for the representation of nodes within the cooperation model, introduced, and sequentially evaluated an algorithm for the formation of the detection group. The taxonomy for cooperation is used for grouping nodes into an interest-based collaborative security system. Additionally, with the formation algorithm, nodes with similar interests as well as property bases can be united into a detection community.

IDSs can definitely benefit from sharing plenty of information. However, some information may not be expected to be exposed to others, for example, IP addresses and logging files. Moreover, the communication among IDSs inevitably increases the traffic of networks and leads to congestion. To address the aforementioned problems, Locasto et al. [2005] proposed a collaborative mechanism for P2P intrusion detection named Worminator. In regards to privacy, the authors used Bloom, a one-way data structure that supports two operations (insertion and verification) to guarantee compactness, resiliency, and security. Regarding limited bandwidth, a network scheduling algorithm is introduced and can dynamically correlate IDSs into a detection community. As it is only a subset of all IDSs, the algorithm can significantly reduce the overhead and mitigate the congestion so that one IDS only communicates with others in the same community.

More recently, Distributed Hash Table (DHT) is widely used to enhance the communication in intrusion detection systems. DHT-based overlay networks can accelerate the network transmission and protect data transmitted via networks. With the peer-to-peer architecture, Marchetti et al. [2009] presented a distributed system in which each collaborative alert aggregator can detect intrusion and disseminate local analysis in a collaborative manner. The system is built on a DHT overlay network, wherein alerts can be quickly shared amongst different nodes. Similarly to Marchetti et al.'s work, Czirkos et al. [2012] proposed Komondor, which used a DHT overlay network named Kademlia. It adopts a peer-to-peer architecture to foster scalability and avoid a single point of failure. Furthermore, Konmondor can minimize the effect of churn[5] caused by the peer-to-peer architecture by remapping keys in each node when a node is leaving and then recalculating the distance to the newly joined nodes.

Summary: According to the analytical framework, the monitoring unit in collaborative intrusion detection is generally an individual intrusion detection system (IDS), and the decision unit is responsible for confidently determining the real intrusions through collaboration. Within the previous section, we placed emphasis on the collaboration unit and shared information. The collaboration unit builds the relationship between different IDSs and shares information about intrusions. After investigating the aforementioned works, we analyzed three highlights of CID, as shown in Table I: communication, robustness, and privacy. Communication is fundamental to collaborative intrusion detection because IDSs need to share security-related information with each other in order to perform a specific task. The mechanism for communication should satisfy both the efficiency and the scalability; that is, nodes should be organized in an effective manner for communication, and network traffic that is generated should be minimized in order to fit into vast networks. Peer-to-peer networks should be the first attempt to enhance the communication in CIDS used in Janakiraman et al. [2003] and Zhang et al. [2003]. Subsequently, overlay networks (e.g., DHT overlay network) are proposed to accelerate the communication in Yegneswaran et al. [2004], Marchetti et al. [2009], and Czirkos and Hosszú [2012], and community-based networks are formed to reduce network traffic and make the communication more effective [Albers et al. 2002;

[5]When nodes join or leave the network frequently, it can cause a fluctuation of the network.

Table I. Highlights in Collaborative Intrusion Detection

Highlight	Items	Literature
Communication	Peer-to-peer	Janakiraman et al. [2003] and Zhang et al. [2003]
	Overlay network	Yegneswaran et al. [2004] and Marchetti et al. [2009]
		Czirkos and Hosszú [2012]
	Cluster formation	Huang and Lee [2003], Albers et al. [2002], and Bye and Albayrak [2008]
		Luther et al. [2007], Kachirski and Guha [2003], and Locasto et al. [2005]
Robustness	CA	Janakiraman et al. [2003] and Yegneswaran et al. [2004]
	Trust/Reputation	Albers et al. [2002] and Locasto et al. [2005]
Privacy	Bloom Filter	Locasto et al. [2005]

Kachirski and Guha 2003; Huang and Lee 2003; Locasto et al. 2005; Luther et al. 2007; Bye and Albayrak 2008]. *Robustness* is the capability of resisting insider attacks which may subvert the entire system. In this section, there were two main methods to ensure the robustness: Certification Authority (CA) and Trust/Reputation. Certification Authorities are brought in for key distribution and authentication. Messages shared in the system can be encrypted or hashed to avoid counterfeit messages, as shown in Janakiraman et al. [2003] and, Yegneswaran et al. [2004]. Trust/Reputation is an alternative option to enhance the robustness. Nodes communicate based on mutual trust in a similar community, as outlined in Albers et al. [2002] and Locasto et al. [2005]. For privacy, bloom filtering is used by Locasto et al. [2005] to protect the sensitive information included in the alerts. Further details of robustness and privacy will be discussed comprehensively and thoroughly in Section 8.

5.2. Collaborative Anti-Spam

The struggle between spam and anti-spam will not likely end in the near future because the Internet is a major tool for advertising and marketing. Loathsome advertisers, virus disseminators, and insidious intruders are attempting to disturb or damage our normal life all the time. For example, they send massive either enticing or boring emails to users whose email addresses are unconsciously exposed in the Internet. Even worse, spam keeps evolving into advanced variants to avoid the detection of traditional anti-spam systems. In this section, we investigate eight emerging anti-spam systems, which improve the accuracy of detection and reduce the risk of infection via collaboration.

SpamNet [Cloudmark 2013] uses a central server model to address these problems. Users can upload their spam into a central server and also can query whether an email is spam or not. But it is no doubt that it has a risk of single point of failure. In such a case, Kong et al. [2006] present a collaborative mechanism for spam filtering. The contributions are twofold: a novel percolation search algorithm, which reliably retrieves content in an unstructured network by looking through only a fraction of the network. It can also avoid single point of failure because all queries and communication are exchanged via email through personal contacts; a well-known digest-based indexing scheme, which can accelerate the process of searching, has high resilience to automatic modification of spam, preserves privacy and produces zero false positives.

Comparing to spam digest proposed in Kong et al. [2006] and Lai et al. [2009] provide an approach of spam rule generation based on rough set theory. They present a collaborative framework to generate, exchange, and manage spam rules. At first, spam rules can be generated in each mail server through rough set theory based on the metadata, for example, header information, keyword frequency and format information. And out-of-date rules are periodically dropped via a reinforcement learning approach. In the sequel, spam rules will be converted into XML format and exchanged by different mail

servers via trusted channels. The limitation of this approach is, however, rough set theory can produce false negatives and false positives, which have been illustrated in the article.

In addition, there are several important challenges (e.g., preserving privacy and retaining important features) when employing collaborative anti-spam systems proposed by Li and Zhong [2008, 2009]. It is no doubt that privacy preservation should be the first and foremost one. Emails may be involved with some private information. If they are published without any preservation, the privacy of participating entities will be exposed and captured by some malicious users. To address these problems, they present a large-scale privacy-aware collaborative anti-spam system called *ALPACAS*. In their framework, anti-spam agents can cooperate by sending or receiving the shingle-based transformed feature set (TFSet)[6] to others to guarantee the confidentiality. However, ALPACAS has two major limitations: it is helpless if there are some malicious email agents who upload erroneous information into the knowledge bases, and it is susceptible to collaborative inference attack in which attackers can infer the content of emails.

Moreover, Sousa et al. [2010] propose a novel collaborative anti-spam system that can be classified into interest-based collaboration. It employs an approach to remove duplicate messages by MD5 signatures of the body messages and sort them chronologically, which is more realistic than random sampling in Zhong et al. [2008]. In particular, in the local view, they use Bayesian filtering to distinguish spam from all emails; from the global prospective, since every email server stores a portion of the spam databases, they can collaborate with each other based on their interests to enhance their accuracy. Nevertheless, it also does not provide an effective approach to solve the problems existing in ALCAPAS.

So far, there are some works that solve the insider attacks using trust and reputation. Sirivianos et al. [2011] introduce the first collaborative spam mitigation system. It takes into account the quality of reports and the social network of reporters' administrators, in order to measure the trustworthiness of the reporters. SocialFilter*,* which they develop can improve the reliability based on Sybil-resilient OSN-based trust inference mechanism. It further enhances the trustworthiness using social links and is able to produce no false positives despite the absence of reports.

Shi et al. [2011] extended the scope of collaboration by introducing three kinds of collaboration for anti-spam systems: (1) *r*ecipients collaboration is that a vast number of recipients can collaborate, share information, and in addition give feedback about whether the email is spam or ham to enhance the accuracy of detection; (2) *honeypots*[7] *collaboration* indicates that spammers that the honeypots glean will be timely shared among email servers; (3) Internet service providers (ISPs)collaboration plays a significant role in spam filtering. With collaboration among ISPs, ISPs can filter or set up a warning to spam in the process of email delivery.

The collaborative security approach is also applied to detect comment spam. The issue of comment spam emerges as the popularity of blogging, in which malicious users want to attach their advertising hyperlinks, malicious or enticing web sites into comments.

PalProtect is proposed as a plug-in of *WordPress*, the most prestigious blogging system all over the world, to collaboratively detect comment spam by Wong et al. [2006]. PalProtect uses other anti-spam plug-ins to perform the detection of

[6]Transformed feature set is the fingerprint of an email (a.k.a. the digests of an email), which can characterize the message content.

[7]From the perspective of anti-spam, a honeypot is a fake email address thst can be effectively used to identify spammers. It is based on the concept that anyone who is not your contact but sends you emails is likely to be a spammer.

spam, and it puts more focus on correlation and sharing information although it also uses its own signature database to categorize and identify the comment spam. It provides five ways to create signature for each comment, of which Z-String is the most remarkable one by counting the frequency of the letters in the comment. Z-String is a one-way data operation, meaning that you cannot construct the original input from the signature, but can still use it as the match object. Therefore, it can reserve the privacy of each comment.

Summary: According to the analytical framework, the monitoring unit can monitor some suspicious emails or comments based on some rules (the rules may be some features of spam). The decision unit will determine them as real spam or not by performing some analysis work. Apart from these two units, collaboration unit and shared information are the mainly parts that we give in the following text.

The difficulties of distinguishing spam and ham in a collaborative manner are various. Spammers always do some tiny alternations to spam in order to escape the inspection of anti-spam systems, which leads to being useless for anti-spam systems to share exact spam. Hence, anti-spam systems instead extract spam patterns based on confirmed spam and disseminate these patterns all over the network to increase the accuracy of spam detection. Nevertheless, it may also produce a high false-positive rate if the patterns are not well abstracted and extracted.

In the literature we investigated, two kinds of extraction techniques are proposed. One is extracting the features of emails, like header information, keyword frequency and format information [Lai et al. 2009]; the other is producing the digests of emails, like shingle-based transformed feature set [Zhong et al. 2008]. These extraction techniques can also help to preserve the privacy of emails, as ham may be very confidential and should not be exposed to other unrelated persons. As mentioned earlier, Bloom Filter and Z-String [Wong 2006] are other two alternative approaches employed in privacy preservation. However, these approaches of privacy preservation, without exception, have degraded the accuracy of detection in certain extend.

Only two of investigated papers have mentioned how to prevent insider attacks to secure the whole system. SocialFilter [Sirivianos et al. 2011] builds a robust relationship between spam detectors via social network. It seems reasonable as friends need to trust each other. But if one is compromised or an essential rantankerous "friend," the performance will be greatly degraded. Another work is from PalProtect [Wong 2006]. Messages are encoded with a Pretty Good Privacy (PGP) key before sent to other systems and each node maintains a "buddy" list to authenticate and decode messages. However, the system requires a lot of maintenance, and the whole system will be easily subverted if one's PGP key is divulged.

5.3. Collaborative Anti-Malware

Conventional anti-malware systems rely on highly trained experts to identify virus, worm, and trojan signatures from binary files [O'Donnell and Prakash 2006]. Collaborative anti-malware systems, which use collaborative filters, can effectively and accurately filter the majority of malware away without too much overhead of individual detection. O'Donnell and Prakash [2006] try to adopt a collaborative approach to detect viruses. As the experimental results indicate, the collaborative filters can increase the speed of detection and extremely low false-positive rate. However, the authors fail to provide more details on techniques when applying collaboration to detect virus.

It is not efficient to store all malware signatures in end-hosts, concluded from the four-month observation of Cha et al. [2011]; Only 0.34% of all signatures in *ClamAV* were necessary for detecting malware. Besides, the current anti-malware systems used to keep all signatures pinned in main memory, which can reduce the performance of

the host, and the matching algorithms are insufficient for high-effective detection. Therefore, they propose an efficient and distributed approach. SplitScreen, the integrated extension to ClamAV, adopts a centralized architecture to reduce the overhead of clients and accelerate the process of malware detection. In an individual domain, the SplitScreen server will distribute the latest malware signatures in the clients. Based on these signatures, the clients can separate suspicious files from harmless ones. After that, they acquire all signatures of suspicious files from the server to identify the malicious files.

Summary: Collaborative anti-malware systems can detect anomalies, viruses, trojan horses, worms, and spyware, and they usually utilize the signatures of malware for the matching process. In the analytical framework of collaborative security systems in Figure 2, the monitoring unit is usually some anti-virus software deployed in a host, and the decision unit can determine if there is some malware running on the host. For collaboration and shared information, message ① is the signatures of suspicious malware in collaborative anti-malware systems, and message ② is the feedback to these signatures from other systems. Similar to collaborative intrusion detection, if one host can individually determine the malware, it will just mark it and disseminate it to other peers. Otherwise, it will ask for other peers or the central server to determine. After all, the database of signatures will be so huge that every single node cannot hold all of them, so the knowledge should be deployed in a distributed way, not only to guarantee the performance of malware detection but also to reduce the storage of signatures in each node.

5.4. Collaborative Identification of Malicious Nodes

Due to easy deployment and low cost, WSN and MANET are ubiquitous to collect either internal or external data for further analysis, for example, identifying malicious nodes. The nodes in these networks, thus, participate in monitoring to provide the evidences of malicious nodes. In order to boost data collection, Cardone et al. [2011] propose a collaborative monitoring system that can bridge these two kinds of networks seamlessly. All the nodes are grouped into different clusters, and one of them is elected as the *root*, then other nodes form a tree-like topology. The data collected in each leaf node is logically transmitted to its parent, eventually the *root*. Therefore, it can obviously facilitate the detection of attacks in system layers, for example, anomalies and viruses. At last, the full assessment and quantitative evaluation in the experiment indicate that the proposed approach is qualified for ensuring effectiveness and feasibility though with limited resources.

The work of Cardone et al. fails to explain the detailed monitoring schedule and the usage of resource. Gu et al. [2012] present an approach to address the traffic-aware monitoring (TRAM) problem. To optimize the usage of the monitoring channels, they come up with three heuristic strategies and additionally develop a TRAM protocol to support the simultaneous operations of monitoring and transmission in mesh networks. Nodes in the mesh network exchange the ID of neighbours' assigned channel, loads, and time allocation to mediate and coordinate monitoring and forwarding to guarantee the maximal monitoring coverage.

AODV is the acronym of Ad-hoc On-demand Distance Vector Routing protocol, which is widely used in ad hoc networks to route and forward packets to the intended receivers. The black hole attack is an important problem that occurs in AODV. Patcha and Mishra [2003] propose a collaborative architecture to detect and exclude malicious nodes that act in groups or alone by extending the watchdog. First, nodes in ad hoc networks are classified into trusted and ordinary nodes. Second, watchdogs are selected from trusted nodes to monitor other nodes (e.g., node energy, node storage capacity

available and node computing power) for a specific period. At last, two thresholds, Suspect and Acceptance, are maintained to determine a compromised or trusted node separately once any node crosses the boundary for all watchdogs' neighbors. The approach is built on the assumption that the network composition is constant, and there are no nodes leaving frequently and rapidly.

Similarly, Krontiris et al. [2009] took sinkhole attacks [Krontiris et al. 2007b] and selective forwarding attacks [Krontiris et al. 2007a] as objectives of prevention. They made a first attempt to formalize attacks and proposed a cooperative algorithm to identify compromised nodes. They made each node participants into identifying the malicious node and providing its evaluation value to neighbours. In the consequence, the approach can produce more accurate results. But it ceases to work if there are many attackers which can launch a collusion attack, and it also can be influenced by dynamic node addition and removal in the networks.

There is also a trend to utilize collaborative approaches in the detection of phishing domains. For example, Zhou et al. [2009], aiming to address the issues of detection of Fast Flux (FF) Phishing Domains, present two approaches to correlate evidences collected from a number of DNS servers and suspicious FF domains. In order to uncover the phishing domains, every node is eager to report the list of suspicious phishing domains. The domains of which the possibility exceeds the threshold are confirmed as real phishing domains. Considering that a centralized architecture is at a risk of single point of failure and insufficient of scalability, they deploy these technologies in the previous work LarSID [Zhou 2007]. LarSID utilizes a publish-subscribe mechanism to share evidences in a peer-to-peer network; not only can nodes share information, but they will also correlate evidences acquired from other nodes.

Summary: Malicious nodes are widely existing in peer-to-peer networks. An individual node lacks of sufficient and necessary evidences to determine the compromised node. Even if they can, it is not guaranteed that they are able to convince others with confirmed nodes. Given that, collaboration among nodes is undoubtedly a better choice. In this case, detection unit in Figure 2 is responsible for reporting dubious nodes against their abnormal behaviors or advices from authorities, and it sends the lists of suspicious nodes (denoted as message ①) to the next unit based on its own knowledge. Collaboration unit is to disseminate its own report and acquire reports from others (denoted as message ②). Some scheme (e.g., MAC) may be exploited to ensure the authentication of reports in this unit [Krontiris et al. 2009]. Decision unit usually utilizes some algorithms to correlate the reports and then decides which nodes are compromised. Threshold [Patcha and Mishra 2003] and majority rule [Krontiris et al. 2009] are two typical approaches found in the literature. The threshold can be derived from the statistics of the specimen or the social theories, that is, experiences undergone before [Patcha and Mishra 2003]. Usually, the identification of malicious nodes is targeting at the routing trap in the host layer and to enhance the robustness of communities.

5.5. Collaborative Malware Detection in Mobile OS

Sophisticated mobile operating systems, from Symbian OS, Windows Mobile to Android, iOS and Windows Phone, have opened up a new era of mobile life. Young as they are, a considerable number of software has been shifted to mobile devices and the number of applications on Mobile OSs has exponentially increased in the past few years. However, malware also swarms into this area and puts a great risk on mobile users. What can ease our worries about the situation is that some techniques and theories of malware detection have been proposed and started to play an indispensable role in mobile times.

SmartSiren, proposed by Cheng et al. [2007], is a collaborative virus detection and alert system for smartphones. In order to eliminate the resource constraints, they utilize a proxy-based architecture, in which every smartphone is only responsible for collecting information of local behaviors and the *proxy* server will carry out a joint analysis in terms of this information for not only single-device but also system-wide detection of abnormal behaviors. It is noteworthy that anonymization and labeling are performed on the reports before submission to prevent privacy from leaking to the proxy server.

Unfortunately, there is a dramatically waning interest in protecting Symbian OS and Palm OS. Android and iOS inversely attract the majority of researchers as well as hackers. Recently, attacks aiming at smartphones are emerging endlessly, like stealing users' sensitive information, making smartphones unavailable and so forth.

Schmidt et al. [2008] propose an approach to monitor and detect collaborative anomalies. The framework can be divided into three parts: on-device analysis, collaboration, and remote analysis. Clients can communicate, for example, sharing detection results or anomalous feature vectors with each other, and are also able to submit data to the remote server once the local detector cannot handle it. Monitoring and Detection in each client is of three-layer architecture. In the lower layer, the main task is to monitor signals or function calls and try to detect anomalies. In the higher layer, collaboration module and response module will take part in forming the collaborative community based on interests. In the sequel, two protocols are provided to either exchange message traffic for a specific computation task or request detectors from its neighbors for a specific event.

Furthermore, Schmidt et al. [2009] have furthered collaborative malware detection, especially on on-device analysis. By performing static analysis on executables, they can obtain their function calls to the Android system. Then, multiple mobile devices sharing the analysis results form a collaborative environment that can effectively enhance the performance of malware detection.

Agarwal et al. [2010] have proposed a collaborative mechanism to diagnose mobile applications in Android and iOS platforms. They first give a brief summary for crash logging mechanisms and analysis of trouble tickets. Then they propose an approach that uses spatial spreading to reduce measurement overhead, statistical inference to recover incomplete data, and adaptive sampling to refine the dependency graph. All these techniques are integrated into a system called MobiBug. The MobiBug server as well as mobile phones that connect to it form a centralized topology. On the phone side, MobiBug matches the crash information in a signature-based manner, and for the failures that are unsuccessfully matched, that is to say potentially new bugs, MobiBug will send them to the server for further probes. In the server side, MobiBug collects massive amount of failure information and conducts dependency analysis and, if necessary, probabilistic analysis to statistically infer incomplete data received.

Oliner et al. [2012] develop a tool, Carat, to perform an energy diagnosis on mobile devices, which can find energy-wasting applications installed on the mobile device. Carat takes a collaborative, black-box approach to find energy bugs in applications. The front-end application collects state information of power usage of applications then transfers them to the Carat server. The back-end analysis engine can statistically analyze the state information stored in the server and return the statistical data (e.g., applications that are using up the battery and whether it is normal, countermeasures) to users. Carat is of a centralized topology and receives state information from thousands of users and returns a customized action list, bug lists and hog lists.

Summary: These types of systems mainly try to mitigate security threats in mobile devices such as privacy leakage, privilege escalation, and resource depletion. There are a variety of characteristics of collaboration in mobile networks, such as the following:

—Due to the inadequate computing ability of the nodes, the majority of collaborative systems employ the centralized architecture, in which all mobile devices send on-device data to the central server.
—Exchanging data can only facilitate individual security detection other than the necessary conditions. Without sharing information, mobile devices can still carry out the detection, although it may be less effective.
—The data are even, if not processed, merely raw. For instance, logs of activities, usage of hardware, and so forth, that is to say, a single mobile device only performs some simple even none analysis. The limited resources have obviously restricted the power and scope of malware detection.
—Current research on malware detection in mobile devices is focusing on privacy preservation because mobile devices store amounts of sensitive information of the owners. It is pivotal to protect privacy information from stealing and tampering. However, hiding some features of information or employing cryptography will degrade the performance of detection. The dilemma does not have an effective solution yet.

5.6. Collaborative Detection and Resistance to Botnets

Botnet is one of the most critical security threats. Botnet is formed by attackers compromising thousands of computers called bots, and attackers control these bots overwhelmingly by sending command and control messages. The bots can be used to steal sensitive information, disseminate spam or virus, and launch a distributed DoS attack. Therefore, collaborative detection and resistance to botnets can fix out security threats as DoS in the network/host layer, malware in the system layers, and the threats in the application layer (e.g., privacy and spam).

In his PhD thesis, Malan [2007] proposes a rapid botnet detection method through a collaborative network of peers. Each node in this network constantly runs software that monitors the behavior of its processes and sends a set of snapshots of those processes' behavior to a snapshot server periodically. By aggregating the snapshots and calculating their similarities across peers, the server can determine which behaviors are anomalous and the purposes of these anomalous behaviors. The architecture adopts a client-server model, which can also bring in the threat of single point of failure.

Wang and Gong [2009] propose a collaborative architecture for detection of botnets. In this architecture, they build an in-depth collaboration of three levels for detection systems, that is, information collaboration, feature collaboration, and decision-making collaboration. It is decentralized, meaning that there is no single point of failure. In different layers of collaboration, different information is exchanged. For example, in the feature collaboration layer, features are extracted and correlated and then sent to each other. However, the paper fails to answer some critical questions like the normalization of information shared among peers, and unknown performance without practical experiments.

As botnets can be automatically evolved as different localized versions in a short period of time, how to find an effective and efficient approach to detect and notify the botnet attack becomes an important and challenging problem. To cope with the problem, Tseng et al. [2011] propose a collective intelligence approach that aims to enable the systematic and dynamic creation of malware information and knowledge. Accordingly, they have developed an anti-botnet platform together with a social networking structure, and an anti-botnet service web site, where the collaborative anti-botnet platform is used to collect the botnet attack information through the honeypot deployment of different organizations and the proposed social networking structure can help build the consensus to select the attributes of the botnet. The collected data can be then sent to the anti-virus software vendor to develop the antidote that can be freely downloaded

by the infected internet users. The paper has elaborately explained the normalization and effectiveness mentioned in the previous work [Wang and Gong 2009].

Although the research on detection of botnets has been conducted for over 10 years, no one can provide comprehensive botnet detection, fulfilling all of the detection requirements and providing a foundation for successful defence against modern botnets. ContraBot is introduced by Stevanovic et al. [2012], which has raised detection of botnets into a systematical level. However, it also has its limitation. The theory is based on the scientific hypothesis that correlating the observations and analyzes from client and network entities will significantly improve the botnet classification ability. Nevertheless, counterfeit observations or fake analyzes, without filtering, may degrade the performance of classification to some extent.

Botnets may range from thousands to millions, it is most likely to lead to congestion of networks when detecting bots. Houmansadr et al. [2012a] propose a low-cost collaborative network watermark to address the aforementioned problem. The approach is implemented into BotMosaic, and it marks command and control messages by inserting a particular pattern into the bots' network traffic; hence, bots are prone to be recognized by clients with much lower cost. In a collaborative community, the impact of the approach can be amplified, which can easily find bots in the network and avert further attacks.

Summary: To sum up, there are two fundamental approaches for botnet detection: detect anomalies in hosts, for example, exposing sensitive information and arbitrarily accessing networks; on the other hand, bots require commands from controller or other peers, which have the similar format and features. So it can help to detect botnets by analyzing network traffics. In the five works we investigated, Malan's work [2007] is typical of detecting anomalies in hosts. The snapshot server gets similarities of behaviors gathered from peers and can decide which behaviors are anomalous and subsequently find the bots. However, BotMosaic [Houmansadr and Borisov 2012a] is impressive to interpolate network traffic and then track the bots by proliferation with collaboration. Wang and Gong [2009], Tseng et al. [2011], and Stevanovic et al. [2012] employ both approaches and largely focus on the collaboration to raise the accuracy of detection. As shown in Figure 2, host anomalies and network traffic (denoted as message ①) can be captured by the detection unit and sent to the collaboration unit for further detection. Nodes in the network are collaborating by sharing their verdicts (denoted as message ②). Using the shared information, the decision unit can carry out further analysis and finally find the bots.

6. TAXONOMY OF COLLABORATIVE SECURITY

The previous section presents a great variety of collaborative security. In this section, we give seven principles for the taxonomy, covering analysis target, timeliness of analysis, architecture, network infrastructure, initiative, shared information, and interoperability.

6.1. Analysis Target

Collaborative security varies from analysis target to detect different attacks and intrusions. In this subsection, we distinguish collaborative security by the source of collected information.

6.1.1. Host Information. Collaborative security systems detect intrusions and attacks by monitoring and analyzing the internals of hosts. It can monitor both dynamic behaviors and static states of the system. The information gathered from hosts can be intrusions [Albers et al. 2002], attacks [O'Donnell and Prakash 2006], or patterns of spam [Wong 2006; Zhong et al. 2008; Lai et al. 2009]. After analyzing and correlating

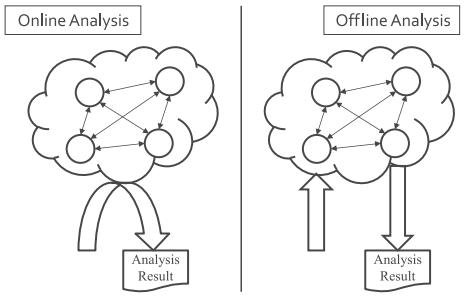


Fig. 3. Timeliness of analysis.

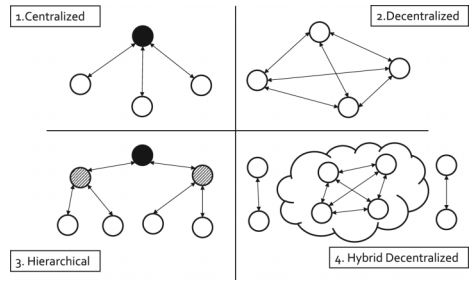


Fig. 4. The taxonomy of architectures.

the information, we can conclude that host information is mainly used for finding out host-aimed attacks (e.g., probing sensitive information in hosts, exhausting the resources of hosts and getting unauthorized permits to some critical components), or helping other hosts better to detect malicious behaviors and be aware of attacks.

6.1.2. Network Traffic. Contrary to host information, another approach is to collect network traffic (i.e., network packets) for detecting malicious activities in the network. The monitor is usually deployed in firewalls or routers and can capture and preprocess the primary packets. The security issues concluded from network packets consist of a blackhole attack in routing [Patcha and Mishra 2003], identification of malicious nodes [Krontiris et al. 2009], and detection of botnets [Houmansadr and Borisov 2012a, 2012b].

6.2. Timeliness of Analysis

In the environment of collaborative security, some systems may need to get the analysis results immediately and take countermeasures against attacks. Nevertheless, the others are deemed to be not demanding for timeliness of analysis results and they do not need to wait for the timely results to execute. It rests on the frequency of attacks, the complexity of detection and capabilities of nodes. More frequent attacks, less complex attacks and more capable nodes can lead to utilizing an immediate analysis and vice visa. The sketch of the timeliness is shown in Figure 3.

6.2.1. Offline Analysis. Offline analysis is more like that the security-related information travels in one way. The initiator sends the security-related information to others and does not need to wait for the results. Afterward, there should be some nodes who take the responsibilities for analyzing the information. In SmartSiren [Cheng et al. 2007], mobile devices are responsible for submitting logs of behaviors to the central server periodically and do not need to wait for the analysis results from the central server. The central server will process, aggregate, and correlate these logs, and detect some potential attacks. In other literature [Wong 2006; Agarwal et al. 2010; Oliner et al. 2012; Stevanovic et al. 2012], offline analysis is conducted while nodes are continuing their job without being blocked by the analysis results.

6.2.2. Online Analysis. In some collaborative security systems, collaborative efforts will immediately turn to analysis results (by synchronization). Nodes that launch a cooperative operation will wait for the analysis results. As in the collaborative identification of malicious nodes, the collaborative operation will not get to its end until they find out the comprised nodes [Patcha and Mishra 2003; Krontiris et al. 2009; Zhou et al. 2009]. The phenomenon also occurs in collaborative anti-spam systems. In ALPACAS [Zhong

et al. 2008; Li et al. 2009], suspicious spam is spread during the collaboration, and the initiator tries to gather the feedbacks from other and finally makes a decision.

6.3. Architecture

Communication and networks are inevitably brought in by collaboration. Peers are connected by some kinds of medium and can communicate to accomplish a specific task. The architecture of collaborative security indicates in which scheme peers are organized and connected and the way they communicate. A summary of architecture is shown in Figure 4.

6.3.1. Centralized. In the centralized architecture, there is usually a central server that is responsible for listening to, communicating with, and ordering peers. Accordingly, the peer-to-peer communication is scarce and restricted. As a consequence, the centralism can benefit global analysis because the central server has overall information produced by each peer. Given that, it can tremendously guarantee the accuracy and correctness of analysis. However, the centralism also produces some side effects: (1) the traffic related to the central server will linearly increase as the number of nodes in the network, thus can inevitably degrade the performance of the central server. In a nutshell, it can decrease its scalability. (2) The centralized architecture is at a high risk of single point of failure. Once the central server ceases to work (e.g., attacked by hackers), the security unit in each node definitely cannot continue to work normally; subsequently, the whole network may undergo heavier attacks and finally crash. The collaborative security systems that use the centralized architecture include Yu et al. [2004], O'Donnell and Prakash [2006], Agarwal et al. [2010], Sirivianos et al. [2011], and Cha et al. [2011]. Especially, malware detection in mobile devices usually employs the centralize architecture shown in Cheng et al. [2007], Agarwal et al. [2010], and Oliner et al. [2012]. The centralized architecture specifies the flow direction of security-related information, that is, transmission of security-related information occurs in the communication channels between the central server and each node. On the other hand, the centralized architecture conveys that the type of security-related information is mainly the raw data or partially processed data.

6.3.2. Decentralized. Disparate with the centralized architecture, the decentralized architecture is of the peer-to-peer form. Every node in this network has the same functions and capabilities; hence, every node plays the same role in collaborative security. Apparently, the decentralized architecture can absolutely avert single point of failure. Furthermore, the autonomy and self-organization make it more scalable. However, the disadvantages of this architecture are threefold: (1) Without a central mediator, distributed nodes will carry only portion of knowledge, which can reduce the accuracy of the detection. (2) The overhead of networks will increase quadratically as the number of nodes. As more and more nodes join the collaboration, the information exchanged among these nodes can be dramatically raised; hence, it may cause high latency of networks. (3) The architecture is, somewhat, influenced by the effect of churn. In an open network, nodes can independently join or leave. In such cases, each node should either renovate its knowledge or calculate the relationship with new comers, respectively; otherwise, security actions may be impacted.

The systems using the decentralized architecture can be found in Zhang et al. [2003], Janakiraman et al. [2003], Zhong et al. [2008], Marchetti et al. [2009], and Krontiris et al. [2009]. The decentralized architecture defines that the flow direction of security-related information is arbitrary and bidirectional among the peers. Since each node is assigned with more analysis work, the exchanged information (e.g., directives and knowledge) will be more fledged and processed.

Table II. Comparison in Characteristics among Different Architectures

Architecture	Accuracy	Scalability	Complexity	Risk of Crash
Centralized	High	Low	Low	High
Decentralized	Low	High	Medium	Low
Hierarchical	High	Medium	High	Medium
Hybrid	Medium	High	High	Medium

6.3.3. Hierarchical. To some extent, the hierarchical architecture is a tradeoff between the centralized architecture and decentralized architecture. It combines centralized and decentralized architectures to remedy respective shortcomings. In a hierarchical architecture, security-related information is collected by the base nodes and transmitted into respective parent. Usually, the flow direction is unidirectional from bottom to up. Taking DOMINO [Yegneswaran et al. 2004] as an example, there are two main kinds of nodes in the architecture, axis nodes and satellite nodes. Axis nodes, the minority of nodes, are pivot because they are the backbone of the architecture. Axis nodes can exchange information peer to peer. Furthermore, they are parents of satellite nodes. An axis node and many satellite nodes form a tree-structured community in which security-related data is always transmitted to the parent. Nevertheless, the challenge in the front of the hierarchical architecture is how to balance the number of nodes in different layers in order to maximize the performance and the effect.

6.3.4. Hybrid Decentralized. The hybrid decentralized architecture is a more complex format of decentralized architecture. In the decentralized network, we divide nodes into several communities under a specific principle. For instance, in CIMD [Bye and Albayrak 2008; Schmidt et al. 2008, 2009], part of nodes are required to form a group that is interest based to refine the function of the group. In Kachirski and Guha [2003], Huang and Lee [2003], and Luther et al. [2007], all nodes are divided into clusters that are distance-based to reduce the overhead of communication among all the nodes. Kong et al. [2006] propose to group nodes based on email contacts, which has already integrated trust in social networks. Moreover, the accuracy of detection and algorithms for formation are the two primary challenges in the hybrid decentralized architecture.

Finally, we present a clear comparison in characteristics (i.e., accuracy, scalability, complexity and risk of crash) of different architectures in Table II. In the table, every characteristic can be assigned with Low, Medium, and High, denoting different levels for each architecture.

6.4. Network Infrastructure

Collaborative security can be used in different networks. In different types of networks, the communication medium and nodes vary a lot. The distinct characteristics of networks can also lead to varied bandwidth or payload capacity. Hence, it will impact the type of exchanged data as well as technologies and algorithms of security detection running in each node.

6.4.1. Wired Network. In the wired network, interconnected nodes have plenty of computing power, storage and high-speed bandwidth. Therefore, physical constraints and polynomial overhead of traffic cannot attract security analysts' attention, and the nodes can perform relatively more complex functions and tasks. Generally, security policies in wired networks are apt to detect sophisticated attacks (e.g., distributed attacks) or filter spam, such as Indra [Janakiraman et al. 2003], DOMINO [Yegneswaran et al. 2004] and Worminator [Locasto et al. 2005]. The main problem for collaborative security in wired networks is how to leverage abundant resources and shared information

to maximize the accuracy of the detection and carry out an in-depth and thorough analysis.

6.4.2. Wireless Network. The nature of wireless networks (e.g., MANET and Wireless Sensor Network (WSN)) makes them susceptible to intrusions and attacks. The characteristics of wireless networks that are attack-prone are fourfold [Zhang et al. 2003]: (1) The electromagnetic signal through the wireless links is easier to be intercepted. Once it is captured by the attacker, it will be likely to cause sensitive information leaks, message contamination, and node impersonation. (2) Mobile nodes that are autonomic and lack adequate physical protection are susceptible to being captured, compromised, and hijacked. (3) Without centralized authority, it may be vulnerable to some attacks, which will disturb the decision-making process. (4) The computing activities are restricted by limited bandwidth, higher consumption, and energy constraints. In addition, disconnected operations and location-based operations both emerging in mobile wireless environment propose a new challenge for collaborative security.

Due to the restriction of MANET networks, the literature [Albers et al. 2002; Huang and Lee 2003; Zhang et al. 2003], as investigated in our survey, adopts some elaborate techniques or shortcuts to reduce the communication and analysis overhead when detecting intrusions. Some other papers [Cheng et al. 2007; Schmidt et al. 2009; Agarwal et al. 2010; Oliner et al. 2012] are concentrating on solving security issues in mobile devices (e.g., detecting malware and buggy applications). Also, many works [Undercoffer et al. 2002; Sarma and Kar 2006; Pathan et al. 2006; Sharma and Ghose 2010] have been found using collaborative security to solve such kinds of attacks.

6.5. Initiative

In this subsection, we divide collaborative security mechanisms into active collaboration and passive collaboration. Nodes in active collaboration may volunteer to execute some security actions with others. The security actions can be predication of one intrusion, identification of a malicious node, or detection of collaborative attacks. On the other hand, nodes in passive collaboration prefer to stay static unless there are some requirements for sending own information (e.g., intrusion and attacks in local database) or receiving new information of intrusion and attacks from others. Also, we can distinguish these two mechanisms based on the shared information. There are more directives and raw (or partially processed) data exchanged in active collaboration because they need to confirm attacks by collaboration. Nevertheless, passive collaboration intends to share less directives and more fledged security-related information to enrich the local knowledge. Most of analysis work is carried out in an individual node; therefore, the communication will be less frequent comparing to active collaboration.

6.5.1. Active Collaboration. In the active mechanism of collaborative security, nodes are eager to contribute themselves to determine intrusions and attacks. As described in Zhang's work [2003], any node that cannot confirm an attack can launch a cooperative way to ask other nodes to give a feedback (i.e., a mere level-of-confidence value to the suspicious attack) about it. Then the host node can calculate based on the feedback to eventually decide whether it is an attack. With the same purpose as Zhang, the collaborative malware detection system in mobile devices proposed by Schmidt et al. [2009] is also in an active mechanism. One mobile device will take a lead to form an interest-based group. Then the members in a group can collaborate to detect some malware existing in mobile devices.

6.5.2. Passive Collaboration. Sharing information and detection of attacks are two stand-alone processes in passive collaboration. The shared information generally consists of latest attack or intrusion updates. Based on the information, local intrusion

detection system can accurately and effectively detect suspicious behaviors or activities. DShield [2013] is a kind of knowledge base from which IDS can acquire intrusion information. The IDSs that just enhance their abilities by synchronizing intrusion information with DShield are acting in a passive manner. Indra [Janakiraman et al. 2003] is a peer-to-peer system that also acts passively. Although the participants would like to share with each other the information of latest intrusions upon detecting them, we still classify it as the passive collaboration because it fails to provide the further analysis. Each node works as a disseminator to send and passively receive security-related information.

6.6. Shared Information

Information sharing is deemed to be the most significant feature of collaborative security. No matter monitoring, analyzing, or decision making, one participant should send information, in a variety of formats, to notify others to perform. Meanwhile, the information has different destinies, either stored as a knowledge base or processed as the input for further analysis. According to this principle, we have categorized shared information in collaborative security into three classes in the following.

6.6.1. Raw Data. Nodes have no ability to determine attacks, spam, or malware; instead, they turn to send raw data gathered by themselves to other more powerful nodes for further analysis. The loss of abilities may be caused by limited resources, deficient knowledge, or tactical consideration. On the other hand, it is no doubt that sharing raw data will worsen the overload of the network and the analysis node due to its redundancy and raise the frequency of exchange due to less filter and process. We have statistically summarized the raw data shared in collaborative security as follows.

—*Suspicious Nodes*. According to Krontiris et al. [2009], nodes share blacklist of suspicious nodes to identify malicious nodes. Similarly, in the Worminator system [Locasto et al. 2005], IP addresses, which are suspected to behave subversively, are reported for identifying attackers. Phishing domains and associated IP address list are exchanged among detection units in Zhou et al. [2009].
—*Suspicious Attacks or Intrusions*. If one behavior occurring in a host or network is detected as a suspicious attack or intrusion, it will be directly shared in the network for identification [Bye and Albayrak 2008].
—*Environmental Data*. Usually, it is collected from physical environments. In MANET and WSN, network overhead and distance between nodes can be shared and used for the formation of clusters to make the system cost-effective in monitoring [Cardone et al. 2011]. On the other hand, the usage of resources, for example, neighbours' assigned channel for monitoring, loads and time allocation, will be shared in Mesh networks with the purpose of reducing the overall overheads. Additionally, Carat [Oliner et al. 2012] collects and shares power usages of different applications for detecting energy bugs.
—*Behavior Logs*. SmartSiren [Cheng et al. 2007] and MobiBug [Agarwal et al. 2010] are both concerned to send information about behaviors logged by mobile phones to the central server, either examining whether there is an attack or determining whether the application has bugs. Especially, the behaviors of applications on mobile phones include sending messages to the network, accessing the inner resources, crash details, and so forth; a snapshot of suspicious behaviors are collected in Malan [2007].

6.6.2. Partially Processed Data. In this case, the capabilities of nodes have been considerably improved, and nodes can exploit available resources to carry out some further analysis. Additionally, there may be some requirements for reducing the amount of

abundant information. As a result, the frequency of exchange is lower than sharing raw data and the data is more organized. After all, nodes are reluctant to face tremendous data, which can definitely degrade their performance. As the result, the data is partially processed before sent to other nodes.

—*Confidence Value*. In Zhang et al. [2003], if one node cannot confirm whether one activity is an intrusion, it will share the state information of this suspicious behavior and wait for the opinions of other participants. In Krontiris et al. [2009], every node will vote for suspicious attackers in order to locate them by calculating the votes. The suspect counter is the basis of calculation for determining attackers which is exchanged among nodes [Pathan et al. 2006].
—*Feature Set*. As presented in Huang and Lee [2003], the feature set can be extracted from one suspicious behavior. Then it is shared for further analysis. The counterpart in anti-spam systems could be a transformed feature set of suspicious emails [Zhong et al. 2008] or *shingle features* of spam and ham [Shi et al. 2011]. In Schmidt et al. [2009], each mobile phone first carries out static ELF analysis then extracts feature vectors from the output to share. ContraBot [Stevanovic et al. 2012] is one of typical botnet detection systems that filters and preprocesses feature data in advance to improve effectiveness and scalability.

6.6.3. Processed Data. Comparing to the first two kinds of data, processed data is the final product produced by security systems. It could be a confirmed attack, an identified malicious node, confident spam, or sheer malware. Since the time of processing data is relatively long and each node can take up most of detection work individually, there is no need to frequently exchange data. Instead, peers only attempt to share the information when necessary. The processed data that are commonly seen in collaborative security is listed as follows.

—*Confirmed Intrusions and Attacks*. In the DOMINO [Yegneswaran et al. 2004] system, every node can summarize the recent attacks and intrusions then share them with others. The same situation can be also found in other systems [Albers et al. 2002; Janakiraman et al. 2003; Luther et al. 2007; Tseng et al. 2011]
—*Alerts/Correlation Results*. Alerts generated in intrusion detection systems are shared as well as the correlation results of them, as described in Yu et al. [2004] and Marchetti et al. [2009]. It can facilitate to find more real and sophisticated attacks, which an individual node cannot afford.
—*Spam*. For anti-spam systems, sharing spam is a straightforward way to filter spam. Spam can be expressed in a variety of formats like spam patterns [Kong et al. 2006], spam rules [Lai et al. 2009], PGP-encoded spam messages [Wong 2006], and spam reports [Sirivianos et al. 2011].

6.7. Interoperability

Interoperability is the ability of collaborative systems to work together with information exchange [Wikipedia 2014]. It defines the mechanism for collaborative systems to communicate, which is either a normalized format for exchanged data, or a communication protocol, or even a complete framework, which describes the communication mechanism between collaborative security systems [Bye 2013]. It is an indispensable but uninspiring feature for collaborative security systems. System designers have to propose a communication mechanism between systems, but oftentimes, they are used to leverage existing standards or frameworks to implement, which is not their main concern. We have investigated and summarized the employed approaches in the literature. For simplification, we classify them into two categories, standard and customized communication. The standard communication means systems employ the de facto standard

Table III. The Statistics of Standard Communication

Standard	Type	Description	System
CIDSS	Data Format	Common Intrusion Detection Signature Standard aims to provide a common data format for intrusion signatures	Bye and Albayrak [2008]
ClamAV	Data Format	It is an open-source anti-virus engine that has uniform and consolidated data format for virus	Cha et al. [2011]
DARPA	Data Format	DARPA owns a public dataset for intrusion detection evaluation	Xu and Ning [2005]
Enron & Bruce Guenter	Data Format	Publicly available email corpuses of both spam and ham	Sousa et al. [2010]
IDMEF	Data Format	Inrusion Detection Message Exchange Format is a standard for data format during the exchange process between IDSs	Albers et al. [2002], Yegneswaran et al. [2004], Duma et al. [2006], Luther et al. [2007], Bye and Albayrak [2008], Pérez et al. [2011], and Czirkos and Hosszú [2012]
IODEF	Data Format	Incident Object Description Exchange Format defines data formats for operational and statistical incidents for exchange	Bye and Albayrak [2008]
TREC & Assassin	Data Format	Publicly available email corpuses of both spam and ham	Zhong et al. [2008]
IDXP	Protocol	Intrusion Detection Exchange Protocol defines the procedure of data exchange between IDSs	Albers et al. [2002]
JXTA	Protocol	Juxtapose is a peer-to-peer protocol specification for collaborative systems to exchange messages	Duma et al. [2006]
DHT	Framework	Distributed Hash Table provides a data storing and quickly lookup service for collaborative systems	Locasto et al. [2005], Marchetti et al. [2009], and Czirkos and Hosszú [2012]
MEET	Framework	Multiply Extensible Event Transport provides a publish-subscribe infrastructure for scalable and effective communication	Gross et al. [2004]
Scribe	Framework	A large-scale and decentralized multicast infrastructure for communication of collaborative systems on application level	Janakiraman et al. [2003]
WordPress	Framework	Plugins on WordPress can setup a channel for different websites to share data	Wong [2006]

in industry to accomplish collaboration, while the customized communication means systems have designed their own specification for communication.

6.7.1. Standard Communication. There are some standard specifications for interoperability between collaborative systems. These specifications have been widely used in industry or academia. For example, the Intrusion Detection Message Exchange Format (IDMEF) is one standard that defines data formats for intrusion information between IDSs [Yegneswaran et al. 2004]. TREC is an email corpus that collects thousands of spam and ham emails and is used for spam detection evaluation [Zhong et al. 2008]. Table III summarizes all standards employed in the surveyed collaborative security systems, consisting of the corresponding specification type, a brief description, and relevant literature.

6.7.2. Customized Communication. Many collaborative security systems employ customized mechanisms to accomplish communication. For instance, Lincoln et al. [2004] design a customized data format for alerts and propose an alert sharing infrastructure for communication between IDSs; [Kong et al. 2006; Sirivianos et al. 2011] propose customized formats for spam features acknowledged and employed by anti-spam systems; SmartSiren [Cheng et al. 2007] is a proposed framework that defines a data exchange format including message content and its hash value, and it provides cheating prevention and privacy protection for collaborative systems.

In addition, there are works that do not mention interoperability. For example, Fung [2011] aims to reveal insider attacks in CIDSs and the significance of robustness. It also proposes mitigations for these insider attacks; Zhu et al. [2012] addresses the incentive challenge generally existing in collaborative systems without describing interoperability in between.

To summarize this section, we list the detailed taxonomy classification for the collaborative security systems mentioned in Section 5 in Table IV. The table covers 44 collaborative systems ranging from 2003 to 2012. The number of each taxonomy are summed for each type of security systems as well as for all systems. In the next section, we will give a comprehensive discussion based on the investigated systems and taxonomies in Table IV.

7. DISCUSSION

This section is devoted to the discussion of the collaborative security systems and taxonomies from three perspectives. First, we build a linkup between collaborative security systems and security threats. Second, we draw conclusions from the observations of Table IV. Third, we try to reveal the relations of different taxonomies. We hope that readers could use these findings to guide the development of future collaborative security systems.

7.1. Linkup with Security Threats

From the investigated collaborative security systems, we identify 10 kinds of security threats, as shown in Section 4, which are prevented or detected by these systems. Table V shows the more detailed correlation between systems and threats. The principles of linking up these systems and threats can be concluded as:

—For general attacks, such as intrusion and malware, we summarize some typical threats from the surveyed literature and the common taxonomy for them (see Section 4). For example, malware has multiple types of malicious behaviors. It may expose users' sensitive information or elevate its privilege to execute malicious code. Therefore, we put all typical threats that malware can cause in this table.
—A botnet comprises of a large number of connected computers, which can launch other attacks in a large scale. Due to its distributed and tremendous features, it can easily launch DDoS attacks and disseminate spam. In addition, stealing users' information is an auxiliary attack, which can quickly collect information for further attacks.

7.2. Observations of Collaborative Security

From the statistics Table IV, we highlight five findings in the following.

7.2.1. Centralized Architecture Dominates in CMDS-MD. Three fourths of the literature we summarized apply the centralized architecture in malware detection on mobile devices. The reason is that a single node cannot independently complete one complicated task; instead, they are usually contributing to collect information or carry out partial work such as filtering out useless information, extracting unique features, and making

Table IV. Statistics of Collaborative Security

Category	System	Target		Timeliness		Architecture				Network		Initiative		Shared Information			Interoperability		
		Host	Network	Off-line	On-line	Centralized	Decentralized	Hierarchical	Hybrid	Wired	Wireless	Active	Passive	Raw	Partially	Processed	Standard	Customized	Unknown
CIDS	Indra	●			●		●			●			●			●	●		
	DOMINO		●	●				●		●			●			●	●		
	Gross et al.		●	●					●	●			●			●	●		
	TRINETR	●	●	●		●				●			●			●	●		
	Lincoln et al.		●	●		●				●			●			●		●	
	Xu and Ning		●	●		●				●			●			●	●		
	Duma et al.	●		●			●			●			●			●	●		
	Luther et al.	●		●					●	●			●			●	●		
	Pérez et al.	●	●	●			●				●		●			●	●		
	Zhang et al.	●		●			●				●	●			●			●	
	Kachirski et al.	●		●					●	●		●				●		●	
	LIDS	●		●			●				●		●			●	●		
	Huang et al.	●	●		●				●		●	●			●			●	
	CIMD		●		●				●	●		●		●			●		
	Fung et al.		●	●			●			●			●			●			●
	Zhu et al.	●	●	●			●			●			●			●			●
	Worminator		●	●			●			●			●			●	●		
	Marchetti et al.	●	●	●				●		●		●				●	●		
	Czirkos et al.	●	●	●			●			●		●		●			●		
Subtotal	19	12	13	16	3	3	9	2	5	15	4	6	13	3	2	14	13	4	2
CASS	SpamNet	●		●		●				●		●				●			●
	Kong et al.	●			●				●	●		●				●		●	
	Lai et al.	●		●			●			●		●				●		●	
	ALPACAS	●			●		●			●		●			●		●		
	Sousa et al.	●		●			●			●		●				●	●		
	SocialFilter		●	●		●				●		●				●		●	
	Shi et al.	●		●			●			●			●		●			●	
	PalProtect	●		●			●			●			●			●	●		
Subtotal	8	7	1	6	2	2	5	0	1	8	0	6	2	0	2	6	3	4	1
CAMS	O'Donnell et al.	●		●		●				●			●			●			●
	SplitScreen	●			●	●				●		●				●	●		
Subtotal	2	2	0	1	1	2	0	0	0	2	0	1	1	0	0	2	1	0	1
CIMN	Ahamed et al.	●		●		●					●		●	●			●		
	Cardone et al.	●		●				●			●	●		●				●	
	Gu et al.		●		●		●			●		●		●				●	
	Patcha et al.		●		●		●			●		●		●				●	
	Krontiris et al.		●		●		●				●	●		●				●	
	LarSID	●			●		●			●		●		●				●	
Subtotal	6	3	3	2	4	1	4	1	0	3	3	5	1	6	0	0	1	5	0
CMDS-MD	SmartSiren	●		●		●					●	●		●				●	
	Schmidt et al.	●			●				●		●	●			●			●	
	MobiBug	●		●		●					●	●		●				●	
	Carat	●		●		●					●	●		●				●	
Subtotal	4	4	0	3	1	3	0	0	1	0	4	4	0	3	1	0	0	4	0
CDRB	Malan et al.	●		●			●			●		●		●				●	
	Wang and Gong	●	●	●			●			●		●		●	●	●		●	
	Tseng et al.	●	●	●		●				●			●			●	●		
	ContraBot	●	●	●		●				●		●			●			●	
	BotMosaic		●		●	●				●		●			●			●	
Subtotal	5	4	4	4	1	3	2	0	0	5	0	4	1	2	3	2	1	4	0
Total	**44**	**32**	**21**	**32**	**12**	**14**	**20**	**3**	**7**	**33**	**11**	**26**	**18**	**13**	**8**	**25**	**19**	**21**	**4**

Table V. The Correlation between Systems and Threats

System	Threats
Collaborative Intrusion Detection	privacy leakage, privilege escalation, authentication violation, denial of service, malicious code execution, abuse of functionality and resource depletion
Collaborative Anti-Spam	spam
Collaborative Anti-Malware Detection (Mobile OS)	privacy leakage, privilege escalation, authentication violation, malicious code execution, abuse of functionality and resource depletion
Collaborative Identification of Malicious Nodes	deceptive interaction and routing trap
Collaborative Detection and Resistance of Botnets	privacy leakage, spam and denial of service

decisions based on own knowledge. In this case, more complicated and time-consuming analysis is conducted by the central server.

7.2.2. Collaborative Security is Badly Needed in Wireless Networks. In particular, collaborative security is badly needed in MANET and WSN, in which bandwidth is relatively low, energy is insufficient, storage is deficient, and computation capability is limited [Zhang et al. 2003; Huang and Lee 2003; Cheng et al. 2007; Oliner et al. 2012]. Thereby, collaborative security in MANET and WSN would focus more on how to remedy issues introduced by resource limitation and improve effectiveness and scalability. However, traditional collaborative security puts it as the key on how to improve accuracy and detect more sophisticated attacks.

7.2.3. Active Is More Popular Than Passive. Active collaborative security can easily attract more analysts' attention because it always takes a lead in actively probing and detecting attacks or anomalies. As shown in Table IV, 26 papers adopt an active mechanism for collaborative security, where the active collaboration has a notable edge on the number. Apparently, it is a more secure mechanism when comparing to passive collaborative security, considering that the active mechanism is to confirm an attack together rather than individually. Passive collaborative security advocates to detecting attacks based on local knowledge. Although it can update its knowledge periodically by acquiring the information of new attacks from else nodes, it still confronts many risks. The loss of abilities of recognizing new attacks (e.g., zero-day) renders the system infectious for a long time. In addition, active collaborative security can effectively find out attacks in advance with innovative techniques, for example, sufficient collaborative analysis, succinct information exchange for increasing performance and scalability and enough detection accuracy for reducing the false-positive rate.

7.2.4. Remarkable Differences of Shared Information in Different Systems. According to our statistics, CIDS, CASS and CAMS systems tend to share processed information. Nevertheless, the share of *raw* information often occurs in CIMN and CMDS-MD systems. The occurrence of the diversity largely depends on the analysis capacity of single node, the timeliness of analysis, and the coupling feature among these systems. Take anti-spam systems as an example. Once an email server receives an email, the server should deliver the email to the specific recipient immediately. Collaboration for discerning spam among multiple servers may lead to a considerable delay. Obviously, the recipient would rather receive a portion of spam than wait many seconds (even minutes) to collaboratively determine whether the email is spam, especially emergency emails. As a consequence, the email server tends to utilize extant algorithms to detect emails based on known spam patterns, which are processed data shared by email servers.

7.2.5. Benefits from Sharing Partially Processed Data. Though the proportion of sharing partially processed data is not very large, it has demonstrated a trend of collaborative security. Most of relevant literature occurs after 2008, and there are some notable advantages [Zhong et al. 2008; Schmidt et al. 2009; Li et al. 2009]: (1) It cannot only address the issues of information redundance with the first option (i.e., sharing raw data) but also be equipped with abilities of being aware of new attacks which are lacking in the third option (i.e., sharing end data). (2) It can help to preserve individual privacy since encryption or hash scheme can be employed in the preprocess, eliminating sensitive information. (3) It can effectively alleviate the pressure of resource for each node, especially in MANET and WSN, without an energy-consuming and in-depth analysis.

7.3. Correlation Analysis between Taxonomies

To have a better understanding of the collaborative system design, it is useful to reveal the (hypothetic) relationships between the taxonomies. For example, we find that the systems with centralized architecture usually conduct an offline analysis. These relationships could be potentially useful when the system designer needs to decide what taxonomies to use.

In this work, we use conditional probability to express these relationships. Conditional probability can illustrate the statistical independence between two categories. In particular, the larger conditional probability is between two categories, the more dependent and stronger the relationship between them should be. Therefore, we can dig out more significant and valuable features for the design of collaborative security systems. Given two categories X and Y of different taxonomies (e.g., centralized and active), the percentage of being of category Y for which are of category X can be obtained by:

$$P(X|Y) = \frac{NUM(X \cap Y)}{NUM(Y)},$$

where $NUM(X \cap Y)$ denotes the number of systems that are both of category X and Y, and $NUM(Y)$ is the number of systems that are of category Y. For example, according to the table, we figure that 73% of wireless systems have employed an active mechanism.

Based on the data in Table IV, we identify correlation values between different taxonomies. And we obtain some interesting observations and selectively draw them in Figure 5, where the size of each node is related with the frequency of occurrence in our survey, and $Y \xrightarrow{p} X$ means $P(X|Y) = p$. According to the figure, we have following highlights:

—Most of systems (86%) with centralized architecture have conducted an offline analysis. Obviously, the central server has abundant collected data and powerful computational resources to carry out some heavyweight analysis.
—A large portion of systems (71%) of hybrid architecture take an active mechanism considering that they usually form several groups and collaboratively make decision. As a security unit, the group in hybrid architecture would like to collaboratively make security decisions by actively sharing information or assigning security tasks.
—Wireless systems prefer sharing raw and partially processed data (82% in total) due to the limited computational resources. Moreover, the active mechanism is the first option (73%) through collaboration among wireless systems. We infer that since nodes in wireless networks lack enough security evidences and computational resources, they turn to launch an active collaboration to make security-related decisions.
—According to our observations, collaborative security systems that take a passive mechanism are often built on a decentralized architecture (50%), sharing processed

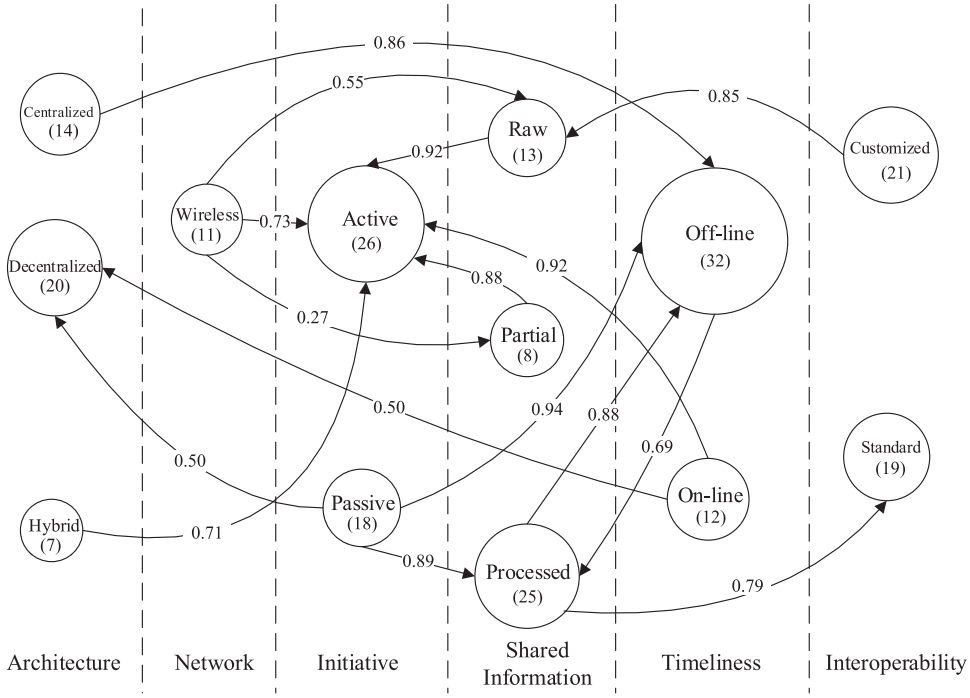


Fig. 5. Relationships between different taxonomies.

data (89%), and conducting an offline analysis (94%). Usually, the systems which take a passive mechanism have a relatively powerful computational ability, and can individually detect attacks. Collaboration means to them more abundant data, specially processed data, for further analysis. In addition, the decentralized architecture guarantees the information can be sufficiently shared between these nodes.

—Sharing raw (92%) or partially processed data (88%) can infer that the system likely uses an active mechanism. It is especially common in collaborative identification of malicious nodes, which share raw or partially processed data and actively find out the attacker.

—Processed data is of a relatively mature format of intrusions and attacks. It can be shared between different security systems in which heavyweight analysis (e.g., correlation analysis) can be performed. Meanwhile, security systems that perform offline analysis usually (88%) share processed data. It makes sense because systems taking offline analysis often carry out a further analysis on known attacks and intrusions. By sharing processed data, each node can acquire enough information for the analysis.

—Online analysis often needs a decentralized topology (50%) and employs an active mechanism (92%). The nodes of online analysis equally play a role in collaborative security, which can lead to a decentralized topology. In addition, as analysis results should be immediately returned to the initiator, these security systems should actively acquire information from each other and conclude a final result.

—In addition, systems employing a customized mechanism of interoperability likely share raw data (85%) for their customized security analysis; and processed data shared between systems usually follows a standard specification (79%) for ease of communication and participancy of other security systems.

8. CHALLENGES

In this section, we summarize five major challenges in designing a collaborative security system. In the surveyed literature, these five challenges are typically mentioned as pivotal aspects to improve and enhance collaborative security systems by many works. Although the aforementioned works may attempt to (partially) solve some of these challenges, there is a need to systematically describe key issues of collaborative security systems and existing approaches. In the following, we will give a detailed description for these challenges, and then provide a schematic summary.

8.1. Privacy

Sharing information is a prerequisite procedure in collaborative security. One node either needs information of attacks and anomalies acquired from others to enrich its local knowledge, or needs to exchange some metadata to complete a detection task. In this case, some confidential information may be leaked unintentionally. Restricting the exposure of information, however, can contradictorily reduce the detection accuracy and increase false positives as well as false negatives. To the best of our knowledge, we have summarized the techniques in these works and categorized them into six classes in the following.

—*Basic Preservation*. Lincoln et al. [2004] solve the problem of privacy preservation in alert correlation. By scrubbing or hashing sensitive fields (e.g., IP addresses and ports), it can protect against privacy leakage. The approach is simply operated; however, it reduces accuracy to a great extent.
—*Concept Hierarchies*. To complement the first approach, Xu and Ning [2005] propose a privacy-preserving alert correlation approach by introducing concept hierarchies. The concept hierarchy is built on the abstraction of alert attributes. Sensitive attributes can be replaced by their upper level concepts. To minimize the uncertainty of generalization, they employ similarity functions to measure the probability of one real attack based on the provided alerts.
—*Bloom Filters* are used in several papers [Gross et al. 2004; Locasto et al. 2005] to preserve privacy during sharing information. Bloom is a one-way data structure that one can hash plain text to avoid, however the reverse.
—*Ticket Exchange* is the measure to protect privacy in SmartSiren [Cheng et al. 2007]. Ticket is the unique identifier distributed by the central server and used to digest the security reports. By exchanging tickets between two nodes with the assistance of the central server (but the central server does not know exactly which the two nodes are), the two nodes can submit reports periodically in an anonymous manner.
—*Z-String* is another one-way data structure employed in privacy preservation [Wong 2006]. It statistically sums up every character occurring in comments and produces statistical results for exchanging.
—*Differential Privacy*, which provides means to maximize the accuracy of queries from the statistical database while minimizing the changes of identifying its records, can also be used to in collaborative security [Reed et al. 2010].

Table VI summarizes the different approaches in privacy preserving. It is worth mentioning that effectiveness and accuracy can be assigned with Low, Medium, and High, denoting different levels for each property. In this investigation, it proves to be critical for collaborative security and the proposed approaches may more or less have some shortcomings in dealing with this dilemma. As a consequence, it raises a problem how to preserve the privacy of users and meanwhile retain the important features of information to guarantee the accuracy of detection. Basic preservation adopts the primary measures to eliminate and remove sensitive information, and it is very effective and

Table VI. Different Approaches of Privacy Preserving

Approach	Literature	Effectiveness	Accuracy
Basic Preservation	Lincoln et al. [2004]	High	Low
Concept Hierarchies	Xu and Ning [2005]	Medium	Medium
Bloom Filter	Gross et al. [2004] Locasto et al. [2005]	High	Medium
Ticket Exchange	Cheng et al. [2007]	Low	High
Z-String	Wong [2006]	High	Low
Differential Privacy	Reed et al. [2010]	Medium	High

simple. However, it removes many important features, which can be very pivotal in detecting or analyzing attacks. Concept hierarchies, as the privacy-preserving measure in alert correlation, is actually an abstraction of sensitive information, for example, using Gateway/Mask representation to represent an IP address. But it requires to refine the abstraction process against false positives or negatives. Bloom filter, as a space-efficient probabilistic data structure, can protect privacy with low false positives, even though it is also restricted by the self-constraints. For example, it does not support modify and delete operations. Ticket exchange is a relatively consuming technique, which should distribute lots of tickets and help to exchange. Z-String is a very simple statistical approach in terms of letters. Effective as it is, it has a relatively low accuracy. Differential privacy can increase the accuracy of queries from statistical databases as much as possible without identifying their records. But the applicable range is relatively small and needs more efforts in the database side. In summary, preserving privacy is still a challenging issue in collaborative security, particularly how to find a good tradeoff between effectiveness and accuracy.

8.2. Accuracy

Accuracy is an essential property of security systems, and the objective where we bring collaboration into security systems is largely to make detection and analysis results more accurate. However, there are mainly two hurdles to affect the accuracy. First, privacy preservation can veil some features so that it will reduce the accuracy. For instance, the application of Bloom Filter will hide the real content of the information, hence can produce false positives. Second, the employed approach to analyze the collected information may vary, especially in accuracy. Unsound and biased criteria of judgement may lead to low accuracy in practice. Therefore, adopting an appropriate approach for privacy preservation and sufficient analysis against the information can increase the accuracy of detection. As a consequence, what to share and how to use are two essential problems in designing and developing collaborative security systems.

8.3. Scalability

To determine whether a collaborative security system is applicable for larger networks with more nodes or not, the following two aspects need to be designed carefully:

—**Communication amongst nodes.** The increment of the number of nodes inevitably causes the overload of networks and longer reaction time since they need to send more information and wait for the response. High-latency networks and inappropriate network topologies will exacerbate this situation. Hence, some overlay networks [Yegneswaran et al. 2004; Marchetti et al. 2009; Czirkos and Hosszú 2012] are utilized to accelerate the communication and reduce the latency of networks, and advanced topologies like hierarchical [Yegneswaran et al. 2004] and hybrid decentralized [Albers et al. 2002; Luther et al. 2007; Bye and Albayrak 2008] topologies are proposed to make the network more reasonable and convenient.

—**Capabilities of pivotal nodes.** In collaborative security systems, the capabilities of some pivotal nodes can directly restrict the scalability of the system. Take Carat [Oliner et al. 2012] as an example. Since all mobile devices will send state information of power usage of applications to the Carat server, the Carat server should have enough capabilities to cope with amounts of information. Otherwise, the performance will be degraded, even the service will be unavailable soon. Therefore, enhancing the capabilities and distributing duties of pivotal nodes can make the system more scalable to some extent.

8.4. Robustness

Robustness means the resilience of collaborative security systems to attacks, especially the insider attacks such as Sybil attack [Douceur 2002], Newcomer attack [Resnick et al. 2000], and Collusion attack [Fung 2011]. Different from the presented threats in Section 4, these attacks are specific to collaborative security, and they take advantage of the provided collaborative mechanism to distribute false information or wrong feedbacks. They may penetrate a collaborative security system acting as "trusted" participants to perform some security-related tasks, which can disturb and obstruct the normal decision making of the whole system. For example, in a Sybil attack, the attacker may create an amount of pseudonymous nodes in order to gain a disproportionate influence. They can spread a rumor in a collaborative security system that one of security systems is compromised, which is actually not, and should be excluded. If the rumor is acknowledged by enough participants, the innocent system is likely excluded, and worse, the whole system can be destroyed gradually. Therefore, collaborative security systems should have a sound mechanism to prevent these kinds of attacks. And the fact is that the insider attacks are often happening in collaborative security systems according to our investigation. Fortunately, some organizations and corporations have proposed several countermeasures against the insider attacks described as follows.

—**Certification Authority (CA).** CA is a special node that is trusted by others in a collaborative security system. It guarantees security and steadiness of the community by distributing keys and certificates to the newly joined and scrutinized nodes. The keys and certificates can be utilized for authentication and encryption of exchanged messages. For example, by exploiting public-key cryptography to create a digital signature for the exchanged messages, it can prevent messages from interpolation and counterfeit, and thereby avoid of insider attacks. As in Janakiraman et al. [2003] and Yegneswaran et al. [2004], the hash values of signatures are appended to the messages using hash scheme and public-key cryptography. A one-way key chain is used to hash messages in Krontiris et al. [2009], where CA should distribute the initial key to each node. However, the main drawback of CA is that it is less scalable and requires more maintenance, for example, key distribution and cryptographic authentication.

—**Trust and Reputation.** Trust and reputation are both used to evaluate the trustworthiness of nodes in a collaborative security system. The difference is that trust comes from subjective and direct experiences with the targeted node, however, reputation is largely based on opinions from other nodes. Nodes with low trustworthiness will not be taken into account for cooperation, and even be removed from the system. Lin and Varadharajan [2006] initiatively set up trust-centric solutions to secure collaboration in mobile agents. They add a trust management layer to collect and evaluate behavioral evidences on top of conventional security layer and facilitate security decision-making process in underlying systems, which is integrated into MobileTrust. A list of acquaintance peers is maintained in Duma et al. [2006] for managing trust. The trustworthiness of a node's neighbours is dynamically

calculated in term of successful experiences and unsuccessful experiences with them. Pérez et al. [2011] propose a collaborative architecture for distributed IDSs with an interdomain trust and reputation model measuring the credibility for each mobile node. The reputation of one moving node is based on the sum of members' experiences with it in the current domain and reputations of other domains. The HIDS with low reputation will not be taken into consideration to detect intrusions. Ahamed et al. [2009] present a novel trust mechanism in wireless sensor network. An enhanced security solution model, Trust-Based Security Solution (TBSS), is proposed to maintain trust relationship amongst the peers. It takes into account both the direct trust (i.e., node's previous experiences with other nodes) and indirect trust (i.e., the group-key and counter values from surrounding nodes) to generate the final trust value. Fung et al. [2009, 2010] address the issue of trust management in collaborative intrusion detection. In Fung et al. [2009], they take the mutual experiences between IDSs as the main reference, and introduce a Dirichlet-based model to quantify the level of trustworthiness. Afterwards, they [2010] propose acquaintance management where each HIDS selects and maintains a list of collaborators. With the collaborative efforts of its acquaintances, one HIDS can benefit from better intrusion detection and assessing the trustworthiness of the acquaintances. By exploiting Bayesian learning, they evaluate both the false-positive rate and false-negative rate of neighbors' opinions and subsequently aggregate them.

As already mentioned, trust and reputation management is prevailing in preventing the insider attacks. As a concept in social science, trust and reputation have been introduced to analyze and evaluate the past interactions of nodes with others. A node may decide whether to accept the invitation of communication from others based on either own direct experiences (i.e., trust) or else indirect comments (i.e., reputation). Due to the effectiveness and practicality, it has been widely used in collaborative security. However, it still leaves some issues. There is lack of a sound criteria and approach to evaluate and quantify the robustness of collaborative security. Some approaches are although proposed, they are usually focusing on some specific insider attacks, however show deficiency against other insider attacks.

8.5. Incentive

Collaboration security is being confronted with an embarrassed situation, where individual systems may sacrifice own CPU/memory and privacy to do some processing work for collaboration. Without a direct benefit, these systems will on balance lose any interest to be involved. Therefore, an incentive for collaboration can effectively raise the enthusiasm of individual systems [Fung 2011]. To the best of our knowledge, there are two kinds of incentive mechanisms applied in collaborative security: (1) coercion incentive, meaning that collaboration is mandatorily performed due to deficient analysis ability and resource limitations, for example, sensor networks that cannot afford traditional consuming security solutions will adopt a collaborative mechanism to make security decisions [Ahamed et al. 2009]; and (2) benefit incentive, which means that collaboration can bring extra benefits at the cost of considerable resources. As in Yegneswaran et al. [2004], the node who shares security-related information has a priority and advantage to recognize the occurrence of intrusions, and logically take a timely measurement to reduce the loss caused by intrusions. Other examples can be found in Cheng et al. [2007] and Reed et al. [2010].

8.6. Correlation of the Challenges

We have picked up some typical works that have (partially) solved the five challenges mentioned in Table VII. Intuitively, most of works (72%) concern about improving

Table VII. Statistics of Challenges

System	Privacy	Accuracy	Scalability	Robustness	Incentive
Indra [Janakiraman et al. 2003]			◓	●	
Gross et al. [Gross et al. 2004]	●	◓			
DOMINO [Yegneswaran et al. 2004]	◓	●	●	●	◓
Lincoln et al. [Lincoln et al. 2004]	●	○	●	◓	
Xu and Ning [Xu and Ning 2005]	●	◓			
Worminator [Locasto et al. 2005]	●	◓	◓		
Duma et al. [Duma et al. 2006]		◓	●	●	
Lin and Varadharajan [Lin and Varadharajan 2006]		◓	◓	●	
Kong et al. [Kong et al. 2006]	◓	●	●	◓	
PalProtect [Wong 2006]	●	○	◓	○	◓
SmartSiren [Cheng et al. 2007]	●	●	○	●	◓
Luther et al. [Luther et al. 2007]		◓	●	●	
Malan et al. [Malan 2007]	◓	●	◓		
LIDS [Albers et al. 2002]			●	◓	
CIMD [Bye and Albayrak 2008]		◓	●		
ALPACAS [Zhong et al. 2008]	●	◓	●	●	
Ahamed et al. [Ahamed et al. 2009]				●	●
Marchetti et al. [Marchetti et al. 2009]			●		
Lai et al. [Lai et al. 2009]	○	●			
Krontiris et al. [Krontiris et al. 2009]			●	◓	
Reed et al. [Reed et al. 2010]	●	◓			
Fung et al. [Fung et al. 2010]		●	●	●	○
Sousa et al. [Sousa et al. 2010]	◓	●	◓	○	
Pérez et al. [Pérez et al. 2011]		●		●	
SocialFilter [Sirivianos et al. 2011]		●	◓	●	
SplitScreen [Cha et al. 2011]	●	◓	●		
Czirkos et al. [Czirkos and Hosszú 2012]		◓	●		
Zhu et al. [Zhu et al. 2012]		◓		●	●
Carat [Oliner et al. 2012]			○		

●: the challenge has been fully addressed
◓: the challenge has been partially addressed
○: the authors have mentioned the challenge but failed to address it
The blank is that the literature does not mention this kind of problem.

accuracy. After all, the target of introducing collaboration is largely to raise the accuracy of detection. In addition, we observe that scalability takes a high weigh (69%) in designing collaborative security systems. The challenge, which is architecture-related, will retain a hotspot topic in this area. Conversely, incentive (17%) does not draw enough attention though it has been proved being facilitating the performance of collaboration to some extent.

To further study the (positive or negative) correlations between the challenges, the system designer can decide what challenges can be handled together if positive correlations exist or given up if negative correlations exist. In this work, we perform the correlation analysis using correlation coefficients between any two challenges as

Table VIII. Correlations of Challenges

	Privacy	Accuracy	Scalability	Robustness	Incentive
Privacy					
Accuracy	–0.70				
Scalability	–0.12	–0.24			
Robustness	0.15	0.29	0.04		
Incentive	NULL	–0.40	–0.52	0.11	

follows. Since we aim to investigate if the relationship between two challenges are loose or tight, conflictive or harmonious, we calculate the Pearson product-moment correlation coefficient between them presented in Table VIII. It provides a measure of linear correlation between two challenges, by giving a value between 1 and –1. According to Table VIII, the designer can clearly learn to leverage the facilitation between positive challenges, and balance negative challenges.

Given two challenges X and Y, the correlation value between them is calculated by dividing the covariance of these two variables by the product of the standard deviations of these two variables. It is worth mentioning that we only take into account the dataset when two challenges both appear in pairs.

The correlation value is in the range of [–1.0, 1.0] (The correlation value of (privacy, incentive) is *NULL* because the standard deviation of variable incentive is zero). The correlation $|r| > 0.7$ reveals a strong correlation; $0.3 < |r| < 0.7$ presents a moderate correlation; and $|r| < 0.3$ presents a weak correlation. In addition, a positive value means a positive correlation and a negative value means a negative correlation.

We have selected several highlights among these correlations as follows:

(1) All the literature that mentions the problem of privacy will also refer to accuracy. According to the correlation value, these two challenges present a strong negative correlation (–0.70), which means that along with privacy is being well solved, the accuracy of collaborative security will be degraded correspondingly. It is reasonable because when sensitive information is sanitized during collaboration, security systems will lose some important information; hence, the accuracy will be reduced.
(2) Robustness is relatively independent with other challenges, of which the absolute values of correlation coefficients are all below 0.3. According to our investigation, workd with the consideration of robustness usually employs an extraordinary mechanism to prevent insider attacks, which is independent with security systems. For example, replying on a trust authority or retaining trust models for its neighbours do not interfere the process of attack detection and consequently will not influence other challenges significantly.
(3) Only 17% of works have mentioned and coped with the incentive and most of them cannot provide an effective solution for this. In addition, it may be surprised that accuracy has a considerable negative correlation (–) with incentive. To some extent, it can imply that although strong incentives can attract more volunteers and efforts, the accuracy is more dependent on analysis methodology and privacy preservation.

9. CONCLUSION

Collaboration in security systems has become a recent trend, with more and more individual systems converting to this method of protection. Compared to traditional individual security, the intention of collaborative security is to share dependable information to provide better security for large systems. This type of security system is more effective and accurate in detecting attacks, with the added ability to detect more sophisticated attacks, such as collaborative attacks. Within this survey, we stated

our motivations to study collaborative security and analyzed many systems equipped with collaborative security, which we supplemented by explaining the advantages and disadvantages of each system. We then provided several comprehensive designs for collaborative security and proceeded to present a thorough discussion of the elements of each design. We laid out several challenges with the current structure of collaborative security systems that have proven to limit the extent of the effectiveness of this type of system. These discussions, as well as a discussion of the trends in collaborative security, provide a platform on which future research on this type of security system can be based.

REFERENCES

Sharad Agarwal, Ratul Mahajan, Alice Zheng, and Victor Bahl. 2010. There's an app for that, but it doesn't work. Diagnosing mobile applications in the wild. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets'10)*. 1–6.

Sheikh I Ahamed, Donghyun Kim, Chowdhury S. Hasan, and Mohammad Zulkernine. 2009. Towards developing a trust-based security solution. In *Proceedings of the 24th ACM Symposium on Applied Computing (SAC'09)*. 2204–2205.

Patrick Albers, Olivier Camp, JeanMarc Percher, Bernard Jouga, and Ricardo Puttini. 2002. Security in ad hoc networks: A general intrusion detection architecture enhancing trust based approaches. In *Proceedings of the 1st International Workshop on Wireless Information Systems (WIS'02)*. 1–12.

Tiranuch Anantvalee and Jie Wu. 2007. A survey on intrusion detection in mobile ad hoc networks. *Wireless Network Security (WNS)* 2, 159–180.

Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. 2012. New privacy issues in mobile telephony: Fix and verification. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)*. 205–216.

Dimitrios Baltatzis, Christos Ilioudis, and George Pangalos. 2012. A role engineering framework to support dynamic authorizations in collaborative environments. *Information Security Journal: A Global Perspective* 21, 1 (Jan. 2012), 12–27.

Elad Barkan, Eli Biham, and Nathan Keller. 2003. Instant ciphertext-only cryptanalysis of GSM encrypted communication. *Advances in Cryptology (CRYPTO)* 21, 3 (March 2003), 392–429.

Michael Becher. 2009. *Security of Smartphones at the Dawn of Their Ubiquitousness*. Universität Mannheim.

Bro. 2013. The Bro Network Security Monitor. Retrieved from http://www.bro-ids.org/.

Rainer Bye. 2013. *Group-based IDS Collaboration Framework: A Case Study of the Artificial Immune System*. Berlin.

Rainer Bye and Sahin Albayrak. 2008. *CIMD-Collaborative Intrusion and Malware Detection*. Technical Report TUB-DAI 08/08-01. Technische Universität Berlin-DAI-Labor. 1–29 pages.

Rainer Bye, Seyit Ahmet Camtepe, and Sahin Albayrak. 2010. Collaborative intrusion detection framework: Characteristics, adversarial opportunities and countermeasures. In *Proceedings of the 19th International Conference on Collaborative Methods for Security and Privacy (CollSec'10)*.

Giuseppe Cardone, Paolo Bellavista, Antonio Corradi, and Luca Foschini. 2011. Effective collaborative monitoring in smart cities: Converging MANET and WSN for fast data collection. In *Proceedings of ITU Kaleidoscope 2011: The Fully Networked Human Innovations for Future Networks and Services (K2011)*. 1–8.

Godwin Caruana and Maozhen Li. 2012. A survey of emerging approaches to spam filtering. *ACM Computing Surveys (CSUR)* 44, 2 (Feb. 2012), 9:1–9:27.

Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G. Andersen. 2011. SplitScreen: Enabling efficient, distributed malware detection. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (USENIX'11)*. 25–38.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3 (July 2009), 15:1–15:58.

Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. 2010. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings of the 31st IEEE Symposium on Security and Privacy (S&P'10)*. 191–206.

Jerry Cheng, S. H. Y. Wong, Hao Yang, and Songwu Lu. 2007. SmartSiren: Virus detection and alert for smartphones. In *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys'07)*. 258–271.

ChinaNews. 2013. Millions of Android Users Are at Risk of Largest-so-BotNet. Retrieved from http://finance.chinanews.com/it/2013/01-09/4474630.shtml.

Chia Yuan Cho, Domagoj Babi ć, Eui Chul Richard Shin, and Dawn Song. 2010. Inference and analysis of formal models of botnet command and control protocols. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS'10)*. 426–439.

Cloudmark 2013. Spam, a Large Collaborative Spam-filtering Community. Retrieved from http://cloudmark.com.

CMU. 2004. CERT. Retrieved from http://www.cert.org/.

Lucian Constantin. 2013. Attackers Are Now Exploiting a Java Zero-day Vulnerability. Retrieved from http://www.computerworld.com/s/article/9235550/Attackers_are_now_exploiting_a_Java_zero_day_vulnerability.

Zoltán Czirkos and Gábor Hosszú. 2012. Enhancing collaborative intrusion detection methods using a kademlia overlay network. In *Information and Communication Technologies (ICT)*, Vol. 7479. 52–63.

David Dagon, Tom Martin, and Thad Starner. 2004. Mobile phones as computing devices: The viruses are coming! *IEEE Pervasive Computing* 3, 4 (Oct. 2004), 11–15.

John R. Douceur. 2002. The Sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS'02)*. 251–260.

Dshield 2013. Dshield. http://www.dshield.org/.

Claudiu Duma, Martin Karresand, Nahid Shahmehri, and Germano Caronni. 2006. A trust-aware, p2p-based overlay for intrusion detection. In *Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)*. 692–697.

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

William Enck, Peter Gilbert, ByungGon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. 2010. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (USENIX'10)*. 1–6.

Adam P. Fuchs, Avik Chaudhuri, and Jeffrey S. Foster. 2009. SCanDroid: Automated security certification of android applications. In *Proceedings of the 31st IEEE Symposium on Security and Privacy (S&P)*.

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

Carol J. Fung, Jie Zhang, Issam Aib, and Raouf Boutaba. 2009. Robust and scalable trust management for collaborative intrusion detection. In *Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM'09)*. New York, 33–40.

Carol J. Fung, Jie Zhang, and Raouf Boutaba. 2010. Effective acquaintance management for collaborative intrusion detection networks. In *Proceedings of the 6th International Conference on Network and Service Management (CNSM'10)*. 158–165.

Michael Grace, Yajin Zhou, Zhi Wang, and Xuxian Jiang. 2012. Systematic detection of capability leaks in stock android smartphones. In *Proceedings of the 19th Network and Distributed System Security Symposium (NDSS'12)*.

Philip Gross, Janak Parekh, and Gail Kaiser. 2004. Secure "Selecticast" for collaborative intrusion detection systems. In *Proceedings of the 3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*.

Qijun Gu, Wanyu Zang, Meng Yu, and Peng Liu. 2012. Collaborative traffic-aware intrusion monitoring in multi-channel mesh networks. In *Proceedings of the 11th International Conference on Trust, Security and Privacy in Computing and Communications*. 793–800.

Amir Houmansadr and Nikita Borisov. 2012a. BotMosaic: Collaborative network watermark for botnet detection. *CoRR* abs/1203.1568, 1–24.

Amir Houmansadr and Nikita Borisov. 2012b. BotMosaic: Collaborative network watermark for the detection of IRC-based botnets. *Journal of Systems and Software* 86, 3 (Nov. 2012), 707–715.

Yian Huang and Wenke Lee. 2003. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'03)*. 135–147.

Nwokedi Idika and Aditya P. Mathur. 2007. *A Survey of Malware Detection Techniques*. Technical Report. Purdue University.

Vineay M. Igure and Ronald D. Williams. 2008. Taxonomies of attacks and vulnerabilities in computer systems. *Communications Surveys & Tutorials (CST)*, 6–19.

Ramaprabhu Janakiraman, Marcel Waldvogel, and Qi Zhang. 2003. Indra: A peer-to-peer approach to network intrusion detection and prevention. In *Proceedings of the 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03)*. 226–231.

Xuxian Jiang and Yajin Zhou. 2013. *Android Malware*. Springer.

Oleg Kachirski and Ratan Guha. 2003. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Vol. 2. 57–64.

Hahnsang Kim, Joshua Smith, and Kang G. Shin. 2008. Detecting energy-greedy anomalies and mobile malware variants. In *Proceedings of the 6th International Conference on Mobile Systems, Applications and Services (MobiSys'08)*. 239–252.

Jungwon Kim, Julie Greensmith, Jamie Twycross, and Uwe Aickelin. 2010. Malicious code execution detection and response immune system inspired by the danger theory. *CoRR* abs/1003.4142.

Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference (CRYPTO'99)*. 388–397.

Joseph S. Kong, Behnam A. Rezaei, Nima Sarshar, Vwani P. Roychowdhury, and P. Oscar Boykin. 2006. Collaborative spam filtering using e-mail networks. *Computer* 39, 8 (Aug. 2006), 67–73.

Ioannis Krontiris, Zinaida Benenson, and Thanassis Giannetsos. 2009. Cooperative intrusion detection in wireless sensor networks. In *Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN'09)*. 263–278.

Ioannis Krontiris, Tassos Dimitriou, and Felix C. Freiling. 2007a. Towards intrusion detection in wireless sensor networks. In *Proceedings of the 13th European Wireless Conference (EWC'07)*. 16.

Ioannis Krontiris, Tassos Dimitriou, Thanassis Giannetsos, and Marios Mpasoukos. 2007b. Intrusion detection of sinkhole attacks in wireless sensor networks. In *Proceedings of the 3rd International Conference on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS'07)*. 150–161.

Gu-Hsin Lai, Chia-Mei Chen, Chi-Sung Laih, and Tsuhan Chen. 2009. A collaborative anti-spam system. *Expert Systems with Applications* 36, 3 (April 2009), 6645–6653.

Kang Li, Zhenyu Zhong, and L Ramaswamy. 2009. Privacy-aware collaborative spam filtering. *IEEE Transactions on Parallel and Distributed Systems* 20, 5 (May 2009), 725–739.

Ching Lin and Vijay Varadharajan. 2006. Trust enhanced security - a new philosophy for secure collaboration of mobile agents. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing*. 1–8.

Patrick Lincoln, Phillip Porras, and Vitally Shmatikov. 2004. Privacy-preserving sharing and correction of security alerts. In *Proceedings of the 13th Conference on USENIX Security Symposium (USENIX'04)*, Vol. 13. 1–17.

Michael Locasto, Janak J. Parekh, Angelos D. Keromytis, and Salvatore J. Stolfo. 2005. Towards collaborative security and p2p intrusion detection. In *Proceedings of the 6th IEEE Information Assurance Workshop (IAW'05)*. 333–339.

K. Luther, R. Bye, T. Alpcan, a. Muller, and S. Albayrak. 2007. A cooperative AIS framework for intrusion detection. In *Proceedings of the IEEE International Conference on Communications (ICC'07)*. 1409–1416.

David J. Malan. 2007. *Rapid Detection of Botnets Through Collaborative Networks of Peers*. Ph.D. Dissertation. Harvard University.

Mirco Marchetti, Michele Messori, and Michele Colajanni. 2009. Peer-to-peer architecture for collaborative intrusion and malware detection on a large scale. In *Proceedings of the 12th International Conference on Information Security (ISC'09)*. 475–490.

Microsoft. 2013. Common Types of Network Attacks. http://technet.microsoft.com/en-us/library/cc959354.aspx

Microsoft. 2014. Account Lockout Policy Overview. Retrieved from http://technet.microsoft.com/en-us/library/cc783851(v=ws.10).aspx.

Markus Miettinen and Perttu Halonen. 2006. Host-based intrusion detection for advanced mobile devices. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)*. 72–76.

MIT Corporation. 2003a. Common Attack Pattern Enumeration and Classification. Retrieved from http://capec.mitre.org.

MIT Corporation. 2003b. Common Vulnerabilities and Exposures. Retrieved from http://cve.mitre.org.

Daniel C. Nash, Thomas L. Martin, Dong S. Ha, and Michael S. Hsiao. 2005. Towards an intrusion detection system for battery exhaustion attacks on mobile computing devices. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom'05)*. 141–145.

Jon Oberheide and Farnam Jahanian. 2010. When mobile is harder than fixed (and vice versa): Demystifying security challenges in mobile environments. In *Proceedings of the 7th Workshop on Mobile Computing Systems & Applications (HotMobile'10)*. 43–48.

Adam J. O'Donnell and Vipul Ved Prakash. 2006. Applying collaborative anti-spam techniques to the anti-virus problem. In *Virus Bulletin*. Montreal.

Adam J. Oliner, Anand Iyer, Eemil Lagerspetz, Sasu Tarkoma, and Ion Stoica. 2012. Collaborative energy debugging for mobile devices. In *Proceedings of the 8th USENIX Conference on Hot Topics in System Dependability (USENIX'12)*. 6–11.

OSSEC 2013. Open Source SECurity. http://www.ossec.net/.

Animesh Patcha and Amitabh Mishra. 2003. Collaborative security architecture for black hole attack prevention in mobile ad hoc networks. In *Proceedings of the 6th IEEE Radio and Wireless Symposium (RWS'03)*. 75–78.

Al-Sakib Khan Pathan, Hyung-Woo Lee, and Choong Seon Hong. 2006. Security in wireless sensor networks: Issues and challenges. In *Proceedings of the 8th International Conference Advanced Communication Technology (ICACT)*, Vol. 2. 1043–1048.

Manuel Gil Pérez, Félix Gómez Mármol, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. 2011. Mobility in collaborative alert systems: Building trust through reputation. In *Proceedings of the IFIP / TC 6th International Conference on Networking (NETWORKING'11)*. 251–262.

Stefan Pütz, Roland Schmitz, and Tobias Martin. 2001. Security mechanisms in UMTS. *Datenschutz und Datensicherheit* 25, 6, 1–10.

Zhiyun Qian, Z. Morley Mao, and Yinglian Xie. 2012. Collaborative TCP sequence number inference attack: How to crack sequence number under a second. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12)*. 593–604.

Radmilo Racic, Denys Ma, and Hao Chen. 2006. Exploiting MMS vulnerabilities to stealthily exhaust mobile phone's battery. In *Securecomm and Workshops*. 1–10.

Jason Reed, Adam J. Aviv, Daniel Wagner, Andreas Haeberlen, Benjamin C. Pierce, and Jonathan M. Smith. 2010. Differential privacy for collaborative security. In *Proceedings of the 3rd European Workshop on System Security (EUROSEC'10)*. ACM, 1–7.

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

Hiren Kumar Deva Sarma Sarma and Avijit Kar. 2006. Security threats in wireless sensor networks. In *Proceedings of 40th Annual IEEE International Carnahan Conferences on Security Technology (ICCST'06)*. 243–251.

Roman Schlegel, Kehuan Zhang, Xiao yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. 2011. Soundcomber: A stealthy and context-aware sound trojan for smartphones. In *Proceedings of the 18th Network and Distributed System Security Symposium (NDSS'11)*.

Aubrey-Derrick Schmidt, Rainer Bye, and Hans-Gunther Schmidt. 2008. *Monitoring Android for Collaborative Anomaly Detection: A First Architectural Draft*. Technical Report TUB-DAI 08/08-02. DAI-Labor der Technischen Universität Berlin.

Aubrey-Derrick Schmidt, Rainer Bye, Hans-Gunther Schmidt, Jan Clausen, Osman Kiraz, Kamer A. Yüksel, Seyit A. Camtepe, and Sahin Albayrak. 2009. Static analysis of executables for collaborative malware detection on android. In *Proceedings of the 8th IEEE International Conference on Communications (ICC'09)*. 631–635.

SecurityFocus. 2003. BUGTRAQ, Security Focus Online. Retrieved from http://www.securityfocus.com/.

JeanMarc Seigneur and Adam Slagell. 2009. *Collaborative Computer Security and Trust Management*. IGI Global, Hershey, New York.

Kalpana Sharma and M. K. Ghose. 2010. Wireless sensor networks: An overview on its security threats. In *IJCA Special Issue on "Mobile Ad-hoc Networks."* 42–45.

Wenxuan Shi, Maoqiang Xie, and Yalou Huang. 2011. Collaborative spam filtering technique based on MIME fingerprints. In *Proceedings of the 9th World Congress on Intelligent Control and Automation (WCICA'11)*. 225–230.

Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, and Qishi Wu. 2009. *AVOIDIT: A Cyber Attack Taxonomy*. Technical Report CS-09-003. University of Memphis.

Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. 2010. Evaluating bluetooth as a medium for botnet command and control. In *Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'10)*. 61–80.

Michael Sirivianos, Kyungbaek Kim, and Xiaowei Yang. 2011. SocialFilter: Introducing social trust to collaborative spam mitigation. In *Proceedings of the 30th IEEE International Conference on Computer and Communications (INFOCOM'11)*. 2300–2308.

SNORT. 2013. Snort. Retrieved from http://www.snort.org/.

Dawn Xiaodong Song, David Wagner, and Xuqing Tian. 2001. Timing analysis of keystrokes and timing attacks on SSH. In *Proceedings of the 10th Conference on USENIX Security Symposium (USENIX'01)*, Vol. 10. 25–25.

Pedro Sousa, Artur Machado, Miguel Rocha, Paulo Cortez, and Miguel Rio. 2010. A collaborative approach for spam detection. In *Proceedings of the 2nd International Conference on Evolving Internet (INTERNET'10)*. 92–97.

Matija Stevanovic, Kasper Revsbech, and Jens Myrup Pedersen. 2012. A collaborative approach to botnet protection. In *Proceedings of the International Cross-Domain Conference and Workshop on Availability, Reliability, and Security (CD-ARES'12)*. 624–638.

Symantec. 2012. *Internet Security Threat Report*. Technical Report 17. Symantec.

Symantec. 2013. *Internet Security Threat Report*. Technical Report 18. Symantec.

Patrick Traynor, William Enck, Patrick McDaniel, and Thomas La Porta. 2006. Mitigating attacks on open functionality in SMS-capable cellular networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom'06)*. 182–193.

Tripwire. 2013. Tripwire, Inc IT Security Software to Improve Data Security and Regulatory Compliance. Retrieved from http://www.tripwire.com/.

Shian-Shyong Tseng, Ai-Chin Lu, Nai-Wen Hsu, Geng-Da Tsai, and Ching-Heng Ku. 2011. Building an anti-botnet platform to mitigate botnet. In *Recent Researches in Communications and Computers*. 409–413.

Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi, and John Pinkston. 2002. Security for sensor networks. *CADIP*.

Jeffrey Undercoffer, Anupam Joshi, and John Pinkston. 2003. Modeling computer attacks: An ontology for intrusion detection. In *Recent Advances in Intrusion Detection (RAID)*, 113–135.

Martin Vuagnoux and Sylvain Pasini. 2009. Compromising electromagnetic emanations of wired and wireless keyboards. In *Proceedings of the 18th Conference on USENIX Security Symposium (USENIX'09)*. 1–16.

Hailong Wang and Zhenghu Gong. 2009. Collaboration-based botnet detection architecture. In *Proceedings of the 2nd International Conference on Intelligent Computation Technology and Automation (ICICTA'09)*. 375–378.

Wikipedia. 2014. Interoperability. Retrieved from http://en.wikipedia.org/wiki/Interoperability.

Benny Wong. 2006. PalProtect: A collaborative security approach to comment spam. In *Proceedings of the IEEE Information Assurance Workshop*. 170–175.

Dingbang Xu and Peng Ning. 2005. Privacy-preserving alert correlation : A concept hierarchy based approach. In *Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC'05)*. 537–546.

Vinod Yegneswaran, Paul Barford, and Somesh Jha. 2004. Global intrusion detection in the domino overlay system. In *Proceedings of Network and Distributed System Security Symposium (NDSS'04)*.

Jinqiao Yu, Y. V. Ramana Reddy, Sentil Selliah, Srinivas Kankanahalli, and Sumitra Reddy. 2004. A collaborative architecture for intrusion detection systems with intelligent agents and knowledge-based alert evaluation. In *Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design*, Vol. 2. 271–276.

Yongguang Zhang, Wenke Lee, and Y. A. Huang. 2003. Intrusion detection techniques for mobile wireless networks. *Wireless Networks* 9, 5 (Sept. 2003), 545–556.

Zhenyu Zhong, Lakshmish Ramaswamy, and Kang Li. 2008. ALPACAS: A large-scale privacy-aware collaborative anti-spam system. In *Proceedings of the 27th IEEE International Conference on Computer and Communications (INFOCOM'08)*. 556–564.

Chenfeng Zhou. 2007. Evaluation of a decentralized architecture for large scale collaborative intrusion detection. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*. 80–89.

Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. 2009. Collaborative detection of fast flux phishing domains. *Journal of Networks (JNW)* 4, 1 (Feb. 2009), 75–84.

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

Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (S&P)*. Washington, DC, 95–109.

Quanyan Zhu, Carol Fung, Raouf Boutaba, and Tamer Baar. 2012. GUIDEX: A game-theoretic incentive-based mechanism for intrusion detection networks. *IEEE Journal on Selected Areas in Communications* 30, 11 (December 2012), 2220–2230.

Received December 2013; revised December 2014; accepted April 2015