University of Nevada, Reno

The Cooperative Defense Overlay Network: A Collaborative Automated Threat Information Sharing Framework for a Safer Internet

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

by

Derek A. Eiler

Dr. Frederick C. Harris, Jr./Thesis Advisor

May, 2014

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

DEREK A. EILER

entitled

The Cooperative Defense Overlay Network: A Collaborative Automated Threat Information Sharing Framework for a Safer Internet

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Frederick C. Harris, Jr., Ph. D., Advisor

Nancy LaTourrette, M.S., Committee Member

Karen A. Schlauch, Ph. D, Graduate School Representative

Marsha H. Read, Ph. D., Dean, Graduate School

May, 2014

Abstract

With the ever-growing proliferation of hardware and software-based computer security exploits and the increasing power and prominence of distributed attacks, network and system administrators are often forced to make a difficult decision: expend tremendous resources on defense from sophisticated and continually evolving attacks from an increasingly dangerous Internet with varying levels of success; or expend fewer resources on defending against common attacks on "low hanging fruit," hoping to avoid the less common but incredibly devastating zero-day worm or botnet attack. Home networks and small organizations are usually forced to choose the latter option and in so doing are left vulnerable to all but the simplest of attacks. While automated tools exist for sharing information about network-based attacks, this sharing is typically limited to administrators of large networks and dedicated security-conscious users, to the exclusion of smaller organizations and novice home users. In this thesis we propose a framework for a cooperative defense overlay network (CODON) in which participants with varying technical abilities and resources can contribute to the security and health of the internet via automated crowdsourcing, rapid information sharing, and the principle of collateral defense.

Acknowledgements

I would like to take this opportunity to thank my family, friends, and educators for encouraging me to pursue higher education and for continually reminding me that few things worth doing are easy.

In particular, I wish to thank my lovely bride Melanie for her steadfast support, prodding, and encouragement amidst the many obstacles that sought to keep me from completing this Master of Science program. I would not be who or where I am without you, Missus.

I would be remiss if I did not thank my good friend Chad Feller for his ongoing encouragement and amusement. Indeed, there are few emotions that cannot be expressed through sausage.

I am also quite grateful for the support and patience of my committee as I worked to refine this thesis and navigate the requisite bureaucracy. Ms. LaTourrette's mentorship proved instrumental in my undergraduate years, and the friendship and accessibility of Dr. Harris, Ms. LaTourrette, and Dr. Schlauch are a great part of why UNR really is a "Tier 1" university.

Table of Contents

List of Figures

List of Tables

1 Introduction

As the internet continues to scale with the increased availability of connectivity in historically less prosperous geographic regions and as everyday objects become network-aware to form the so-called “internet of things” [5], we believe cooperative defense models that encourage “good neighbor” information sharing will be crucial to improving the safety of the internet as a whole. While automated tools exist for sharing information about network-based attacks, this sharing is usually limited to administrators of large networks and dedicated security-conscious users, to the exclusion of smaller organizations and novice home users. In this thesis we propose a framework for a cooperative defense overlay network in which participants with varying technical abilities and resources, but particularly those with minimal technical ability, can contribute to the security and health of the internet via crowdsourced blacklist generation, peer-to-peer security information sharing, and the principle of collateral defense.

In Chapter 2 we explore prevalent network-based distributed attacks against networked systems, current and proposed methods of defense against distributed attacks, and related work. We then illustrate the need for a cooperative and collaborative defense that extends beyond a small alliance of large and well-resourced organizations. In Chapter 3, we describe the architecture and functionality of our proposed cooperative defense overlay network (CODON) as well as anticipated barriers to adoption and potential watershed moments that may spur adoption by large portions of the internet community. We conclude this thesis with a brief discussion of our conclusions and future work in Chapters 4 and 5, respectively.

2 Background and Related Work

Before discussing the need for and value of implementing our proposed CODON framework, we will first examine various threats, both ubiquitous and less common, against which computer network and system administrators, referred to collectively in this thesis as "system administrators," must defend in the course of maintaining the health and security of their various systems. We will also discuss relevant computer security concepts, best practices employed to defend against large-scale distributed threats, and work related to CODON. It should be clear by the end of this chapter that the internet needs defensive options capable of scaling as easily as common threats already do.

2.1 The Internet

Thousands of independently-managed computer networks known as autonomous systems (ASes) across the world connect to form a vast and decentralized "network of networks" known as the internet. The internet lacks a central point of authority governing its operation, and owes much of its success and growth to cleverly-designed and evolving protocols that enable different ASes to interoperate despite their technological, political, and ideological differences. These protocols generally, but imperfectly, allow the internet to be resilient to localized system failures and require minimal state to be maintained by each AS, in many cases adapting to topology changes with little or no human interaction. In general, systems and services can join the vast internet without requiring the rest of the internet to first undergo costly upgrades, re-architecture, or widespread manual intervention. Decentralized services that support the internet such as the Domain Name System (DNS) require a degree of centralized management for human convenience, but data can generally flow between distant networks in the absence of central overarching management. A key incentive for the various ASes to

cooperate is the concept of “fate sharing”: various ASes and the internet as a whole will succeed or fail together, so it is generally in an AS’s best interest to cooperate with others to ensure its own survival. Fate sharing requires a shared goal and a degree of implicit trust between ASes, and this is simply not enough to ensure that malicious AS administrators [61] or the coordinated effort of rogue systems inside otherwise well-behaved ASes will not affect internet operations in undesirable ways.

2.2 Overlay Networks

An overlay network is a subset of nodes in a network that functionally form another network; in other words, an overlay network is a network within a larger “underlay network” whose topology may have little or no similarity to the underlay network. Consider the example of a four-node overlay network of computers running a simplified peer-to-peer file sharing software as illustrated below in Figure 2.1.

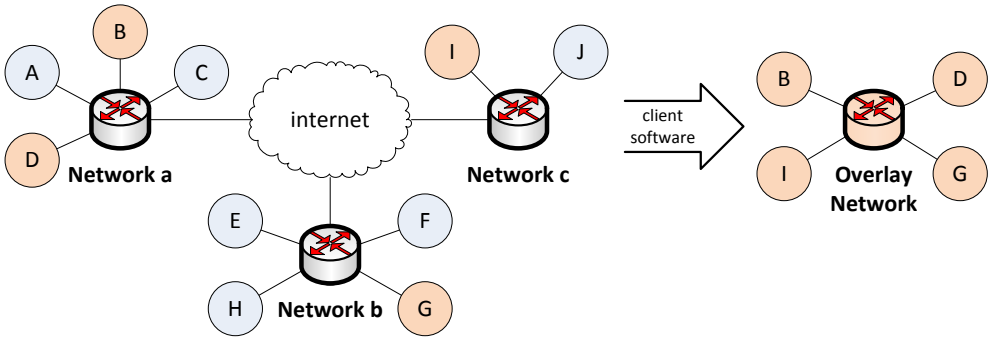

Figure 2.1 Nodes B, D, G, and I on geographically and administratively distinct networks connect via software to form an overlay network. The nodes then communicate with one another as though they are on the same network.

In this example overlay network, the four nodes B, D, G, and I, behave as though they all reside on the same local network, while two of the nodes (G and I) actually reside on different continents with numerous ASes separating them. When a node in the overlay network wishes to search for a particular file or share some useful information via peer-to-peer file sharing

software, it will broadcast the search request to its neighbors in the overlay network while remaining functionally unaware of neighbors in the underlay network.

Overlay networks are commonly formed via the use of peer-to-peer software, as the example above suggested. At its simplest, peer-to-peer software is software that enables multiple computers on a network to communicate directly with one another with minimal, if any, reliance on dedicated and stable centralized servers. Peer-to-peer file sharing software such as BitTorrent [13] has been used to cheaply distribute entire operating systems in parallel amongst technical users as in the case of Debian Linux and many others [64], to distribute enthusiast-generated multimedia content amongst niche online communities [51], and even to distribute software updates for mainstream commercial video games [9]. In the case of BitTorrent, a “torrent” file is used to describe the shared file and to direct clients to a “tracker”. The tracker is a server that tracks other clients, or “peers,” with partial or full copies of the data described in the torrent file, though it is possible to discover peers by other methods, but the tracker does not distribute the shared file. File data is typically subdivided into many small chunks to allow for fast parallel downloading from multiple peers that may not yet have a full copy of the data. A peer with a full copy of the desired data is known as a “seed,” and a high seed-to-peer ratio ensures quick distribution of the data to new peers. A key disadvantage of peer-to-peer file sharing software such as BitTorrent is that the quality of service can fluctuate greatly with peer churn in the network, or “swarm,” as many participants join only long enough to download their desired files and quickly leave the swarm to conserve their own resources such as network bandwidth. Without enough seeds in the swarm or with too much churn, it becomes possible that a peer may not be able to collect all the chunks shared throughout the swarm to assemble a full copy of the original data.

Peer-to-peer file sharing also places inherent trust in fellow swarm participants who may claim to host legitimate content while actually providing no data or malicious data. Hashing can be used to validate the integrity of data distributed by participants, introducing a slight performance impact on participants as the hashing process and the hashes themselves require additional effort to generate, distribute, and verify, but the trust issue remains. For example, an attacker could share a virus purporting to be a recording of a musical performance and publish the hash of that virus. Participants who download the virus can verify that the file has not been modified during distribution over the peer-to-peer network by hashing the downloaded chunks or the complete file, but they may not be able to verify the validity of the content, i.e. determine that the file is really a virus, until a particular chunk or in some cases the entire file has been downloaded. We illustrate this problem below in Figure 2.2 using the example of a large file that has been tampered with and subsequently shared via a peer-to-peer network.

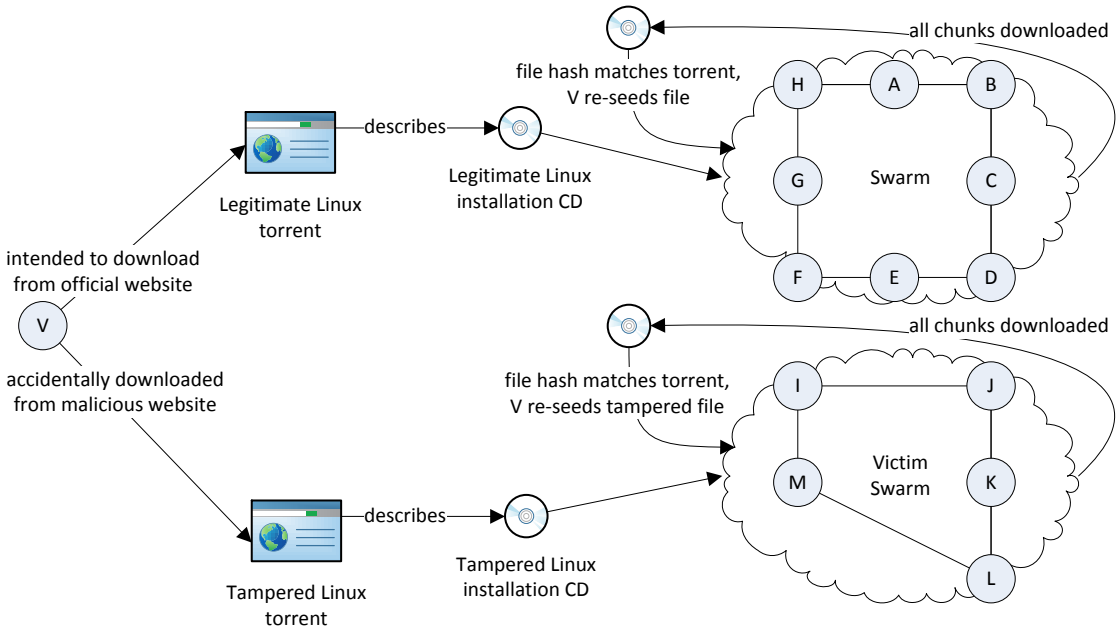

Figure 2.2 Victim V downloads a tampered Linux installation DVD via BitTorrent. Hashing the file downloaded from the swarm shows that the file was not tampered in transit, but the end result is still untrustworthy because the file was tampered before entering the swarm. It is difficult to verify that the file has the desired contents until the download is complete.

The Skype [62] audio/video conferencing system uses a hybrid peer-to-peer and client-server model, initially leveraging end-user resources to reduce the burden placed on expensive centralized server resources. As it has grown in worldwide popularity and server resources have become less expensive, Skype and its parent company Microsoft have shifted much of the computing burden toward centralized servers known as “supernodes” and removed the ability for Skype users to participate as supernodes [27,28]. An important advantage of this hybrid approach is that the overlay network is more resilient to participant churn as many users join and disconnect from the network. This has become increasingly common as users shift away from running the Skype client software on desktop computers in favor of laptops and other portable devices that have additional resource constraints such as limited battery life and intermittent network connectivity. Skype’s change from a primarily peer-to-peer architecture to the more traditional client-server architecture has been criticized due to the potential loss of privacy caused by temporarily storing call-related data on company-owned servers.

2.3 The Security Triad

The concepts of confidentiality, integrity, and availability are widely discussed in the field of information security, and are commonly referred to collectively as the security triad or the mnemonic “CIA triad.” This triad is pertinent to almost every resource an administrator might seek to protect, and network-based attacks seek to subvert at least one of the three elements of the triad. Regulatory requirements and standards [32] often require that specific elements of the security triad be addressed to protect particular categories of sensitive information. The security triad can be summarized as follows, with examples provided below in Table 2.1.

Table 2.1 Common threats against an e-commerce web server and corresponding safeguards

Threat	Threatens Availability, Integrity, or Confidentiality?	Typical Safeguards
SQL injection via "product search" web form could modify or divulge sensitive data	A, I, C	Perform input validation and sanitize form field contents before performing query on back-end product database. Do not grant write/modify privileges to DB account used by web server.
Sudden high volume of shoppers overburdens server	A	Use a load balancing system to distribute requests across multiple web servers. Optimize system software and architecture to minimize resources consumed when processing requests and serving results.
Software glitch causes existing customer data to be overwritten when a new customer makes a purchase	I	Require changes to software to be reviewed and approved by management before implementation. Perform thorough testing on separate testing or "staging" servers before installing software on production servers.

Confidentiality entails ensuring that a sensitive resource is disclosed only to its intended audience(s), commonly through the use of data encryption or strict access controls. Integrity is the protection of a resource against unauthorized or undesired changes, including deletion or corruption, and includes proving whether a resource was tampered with via the use of audit logs, file hashing, and encryption certificate-based digital signatures. Availability is ensuring that a resource continues to function as designed and can be both successfully and timely accessed when needed. Though an administrator is often concerned about safeguarding all three elements of the security triad for a resource on the network, there are instances in which one or two elements may not be important; for example, the message contents of a severe weather alert warning system may by its very nature not require confidentiality safeguards, whereas the message's integrity and availability may be of paramount importance.

2.4 Asymmetric and Symmetric Key Cryptography

Put simply, encryption is the process of modifying a message in such a way that only its intended recipient(s) may understand it, protecting the encrypted message's content from being divulged to a third party if intercepted. We refrain from presenting an exhaustive explanation of cryptography in this section, as it is a field of study unto itself. Instead, we present a high level overview of the encryption process and briefly discuss the elements of cryptography pertinent to our work: symmetric and asymmetric key cryptography. The interested reader is referred to [3] for a more thorough coverage of cryptography.

In cryptography, the original unencrypted message is referred to as "plaintext." Plaintext is encrypted using a carefully devised function known as a "cipher," and the resulting encrypted message is known as the "ciphertext." Ciphers use plaintext and a "key" as input, where the key is usually a secret string that is unfeasible to correctly guess or compute given the ciphertext. A good cipher similarly makes it unfeasible to correctly guess or compute the plaintext given the ciphertext. There are numerous ciphers of varying complexity, and many are available for public use and analysis while many others remain closely guarded secrets. If an attacker is able to intercept ongoing encrypted communication and is also able to determine the cipher and key used for decryption, it is often a trivial task to derive the plaintext, thereby defeating the encryption.

For an encryption algorithm to be useful, the intended recipient must be able to convert the received ciphertext into plaintext. Therefore, both the sender and the intended recipient must agree in advance upon a cipher and use appropriate key(s) for encrypting and/or decrypting communication. Cryptographic algorithms may be divided into two classes based on their use of keys: symmetric key cryptography, in which the sender and receiver use the same

key to encrypt and decrypt all messages; and asymmetric key cryptography or “public key cryptography,” in which messages are encrypted using a publicly available key and decrypted using a privately held key. Symmetric key ciphers are generally faster than asymmetric key ciphers and symmetric keys tend to be stronger than asymmetric keys for a given key length, but symmetric ciphers require that both parties agree on a shared secret in advance and this can be quite burdensome. Asymmetric key cryptography tends to perform more slowly than symmetric key cryptography but allows many parties to encrypt messages to the same recipient using a publicly available key.

It is possible for two parties to establish a shared secret key in such a way that a third party eavesdropping on communications between them cannot determine the agreed upon secret without significant computational effort, and this is frequently done via the Diffie-Hellman key exchange protocol [10]. Unfortunately this exchange is subject to a “man in the middle” attack wherein an attacker could both intercept and modify communications between the sender and the recipient and cause them to unwittingly establish shared secrets with the attacker instead of with one another. Asymmetric key cryptography removes the need for each party to establish a shared secret to communicate with one another securely. The de facto standard for asymmetric key cryptography is RSA, in which a party generates a public key and a private key that are mathematically related. Although the keys are related, when they are properly generated it is extremely difficult to derive the private key given the public key since there are no known algorithms for efficiently factoring arbitrarily large numbers. The public key is made available for anyone to use when encrypting communication destined for the key pair’s owner, and the private key is the only key capable of decrypting the communication. The Diffie Hellman key exchange and use of RSA key pairs are described in more detail in [20] and [21].

To take advantage of the generally faster performance of symmetric key encryption, avoid the need to agree upon a shared secret offline, and work around the man in the middle problem, it is common for the sender and receiver to use asymmetric encryption during the session establishment process, as in the case of the Transport Layer Security (TLS) protocol [18]. Within this encrypted conversation, both parties agree upon a symmetric key and subsequently use a symmetric cipher to encrypt their communications. When using a symmetric cipher, parties often choose to establish a replacement key, or to "re-key," once a predetermined time has elapsed or a certain number of bytes have been transferred. This reduces the amount of time available to an attacker to correctly guess the symmetric key and in turn decrypt the ongoing conversation.

2.5 Vulnerabilities, Threats, Exploits, Safeguards, and Remediation

Vulnerabilities are weaknesses or defects that make it possible for attackers, or even inept administrators and users, to render a system inoperable or to cause the system to perform undesirable actions not intended by the creator or administrator. Remediation is the process of fixing a particular vulnerability and/or recovering from an aftermath of an exploit. Exploits are software tools or other mechanisms of taking advantage of a particular vulnerability. The most effective exploits are those for which no remediation actions have yet been developed or published, typically because the corresponding vulnerability is not widely known or is difficult to quickly remediate due to cost or complexity. Such exploits are known as zero-day exploits.

Widely-used and emerging standards and technologies exist for describing and sharing information about computer vulnerabilities, exploits, and remediation actions. These include MITRE's Common Vulnerabilities and Exposures® (CVE®) [40] and Open Vulnerability and Assessment Language® (OVAL®) [41], as well as the US Department of Commerce's National

Institute of Standards and Technology (NIST)'s effort to synthesize and automate the use of computer security standards via the Security Content Automation Protocol (SCAP) [43]. A promising future resource for large-scale remediation is NIST's emerging Common Remediation Enumeration (CRE) standard [42], which defines remediation steps in an automation-friendly format as well a framework for distributed remediation publishing.

Two commonly deployed software tools used to detect running software and related known vulnerabilities are Nessus [69] and nmap [38]. Such software is used by system administrators and attackers alike to detect vulnerabilities for the purposes of proactive threat remediation and reconnaissance, respectively. And while several commercial tools such as Nexpose [56] exist to summarize vulnerabilities and track remediation steps, these tools are by their nature ill-suited for detecting actual attack activity.

2.6 Intrusion Detection and Prevention Systems

Many organizations and some home users employ intrusion detection systems (IDS), intrusion prevention systems (IPS), and/or intrusion detection and prevention systems (IDPS). An intrusion detection system (IDS) passively detects malicious or suspicious behavior and only logs the event for future reporting. Unlike an IDS, an IPS can perform reactive functions such as disconnecting and preventing network communication between the parties involved in suspicious activity. An IDPS performs the functions of both an IDS and an IPS, but the distinction between an IPS and IDPS is somewhat nebulous and IDPS tends to be the favored term. The most commonly used and well-regarded open-source IDPS is Snort [11], although the multithreaded Suricata [48] IDPS has arisen as a new contender with performance features like multithreading and experimental graphics card acceleration support.

An IDS can rely on detection of statistically anomalous behavior that varies from an established baseline in an organization, as well as known behavior patterns or “signatures” to detect or prevent undesirable behavior. Establishing a baseline of what is considered normal activity on a network requires the administrator to understand what normal really means, or should mean, for the network in question. This baseline can become skewed if unusual activity occurs while the baseline is being established. With very large networks and with home users it can be difficult to establish a baseline due the wide range of activity that legitimately occurs on the network.

Unlike statistical behavior based detection, signature based detection is not tailored to a particular network and the signatures can easily be shared and used by IDS administrators throughout the internet community. Nevertheless, great skill is required to create an efficient and precise signature that detects undesirable behavior with no or minimal false positive detections. Creating signatures requires careful analysis of the raw network communication or “packet captures” between the misbehaving host and the victim, which can be further complicated by the ubiquity of firewalls, network address translation, and other hosts that legitimately modify network communication between hosts. Packet capture analysis is further complicated by the existence of multiple hosts communicating over the same network simultaneously, making it difficult to isolate communication strictly related to the suspicious or malicious behavior. And while signatures are widely distributed online, file size and concerns about divulging sensitive internal communications and private network topology cause packet captures to be less commonly shared for public analysis.

The use of an IDS inside a very large organization’s network still provides a very limited view of malicious behavior on the internet, making it difficult to identify whether the behavior is

part of a larger scale attack affecting multiple organizations. Few organizations have a view of large geographical regions, so the collection of very large scale network traffic data is left to transit providers, content distribution networks (discussed briefly in Section 2.9.3), and research organizations with whom the former are willing to share useful data samples. Public access to large scale attack data is very limited, typically consisting of select researchers' analysis rather than the actual data.

2.7 Firewalls

Firewalls are network security systems commonly implemented at the edge or border of a network to separate a network from other networks outside the administrator's control. Firewalls are also used to separate multiple networks within a larger network controlled by the same administrator based on organizational or regulatory security requirements. Firewalls are commonly implemented as dedicated network hardware appliances, as software installed on a computer as an add-on package, as an operating system feature. Firewall functionality is usually embedded into the firmware of the "all-in-one" multifunction routers leased or rented by many ISPs to their customers. Ingress and egress control decisions are based on sets of rules configured on the firewall and may be as simple as allowing or denying communication between two IP address ranges and a particular port number – for example, "allow traffic from 10.0.0.0 via TCP port 80 to 192.168.0.0 via any port." More complex rules may use multiple criteria, and sets of rules may be chained together to enforce finer-grained control over traffic. While any firewall can filter traffic based on source and destination IP addresses, subnets and ports, more advanced firewalls use stateful packet inspection or even application-layer protocol inspection to make more intelligent access control decisions based on session state and application behavior. Firewalls provide logging mechanisms that can usually be configured to adjust the

verbosity and level of detail and may produce a large volume of log data under load. This log data can prove very useful for troubleshooting and security event correlation when used in conjunction with an IDPS or specialized log correlation software, but storage and processing constraints tend to limit the amount and detail of log data retained by smaller organizations.

2.8 Honeypots and Honeynets

Honeypots are systems used by researchers and various organizations which have been made intentionally by the administrator to be vulnerable to known threats. By allowing an attacker to compromise a honeypot and access it for some period of time instead of preventing the attack with an IDPS or other tools, the administrator can observe an attacker's subsequent activity even down to the keystroke and gain insight into the attacker's behavior and the tools used by the attacker. In practice, the ubiquity and decreasing cost of server virtualization makes it possible for an administrator to configure a honeypot, allow it to be compromised, and later revert the honeypot to an uncompromised state with very little effort. It is very important that the attacker believe the honeypot is a legitimate system, and toward that end honeypot systems such as Honeyd [55] have been developed which mimic even the nuanced behavior of various operating systems' TCP/IP stacks. A honeynet is a collection of honeypot systems working together in concert to provide a more thorough observation of an attacker's activity or simply waste an attacker's time and resources by fooling the attacker into believing the honeynet contains useful information to steal. Similar to a honeynet but much less interactive, a "network telescope" or "darknet" is a system that observes activity targeting unused IP addresses under the assumption that legitimate traffic would not attempt to access those addresses, as in the case of the UCSD Network Telescope [14]. The USCD Network Telescope uses a very large and mostly empty block of publicly reachable IP addresses (a /8 block or roughly 16,777,216

addresses) that is passively monitored for malicious activity and provides the logged data to select security researchers for analysis. Honeypots and honeynets can provide valuable information about emerging network-based attack behaviors and trends, whether distributed or more localized. An important aspect missing from traditional honeypot systems is end-user or client interaction, as they generally focus on observing how an attacker interacts with a server or service rather than how an end-user interacts with an attacker and the attacker's ensuing response to end-user stimulus. In response to this shortcoming, "client honeypot" systems such as PhoneyC [45] have been developed which mimic end-user behavior such as browsing the web for executable files or simulating a user opening links included in unsolicited emails. Both traditional and client honeypot systems are well-suited for use in conjunction with the CODON Sensor software that we describe later in Section 3.3.3.

2.9 Distributed Attacks

The internet's oft-used nickname of "information superhighway" is appropriate: it allows a very large number of people to access distant destinations quickly, whether with good or evil intent. A distributed attack is an attack against a system via multiple attacking nodes that cooperate in an attempt to inflict greater damage or increase the probability of a successful attack than is possible with a single attacking node. Such attacks leverage the power of the internet's decentralized and scalable nature for harm rather than good. The internet is designed to allow participants to easily and quickly communicate with one another, not to hinder communication, and this opens many avenues for malicious participants to wreak havoc.

2.9.1 Distributed Denial of Service and Botnets

The most well-known distributed attack is the distributed denial of service (DDoS) attack. In this attack, a large number of systems work in concert to overwhelm a victim's system

with service requests or other network traffic. These requests are often incomplete or malformed in such a way that the victim is forced to expend much more effort responding to the requests than the attacker expends sending them [72]. As the victim attempts to process and respond to each incoming malicious service request, the victim quickly becomes unable to respond to legitimate requests in a timely manner, sometimes even stopping completely under the enormous load. There are many variations on the DDoS attack tailored to different applications and network protocols. In a so-called reflection attack, attacking hosts send forged requests containing a victim's IP address in the sender or source IP field to innocent servers that are configured to respond to incoming requests. This causes the innocent servers to "reflect" a large amount of traffic to the victim while obscuring the real attacker's identity [72]. DDoS attacks can be difficult to distinguish from legitimate so-called "flash crowds" of users attempting to access a system simultaneously due to a sudden growth in the system's popularity [53]. We illustrate a successful DDoS attack based on a victim's network bandwidth below in Figure 2.3.

Participants in a DDoS attack are usually part of a network of compromised systems known as a "botnet." Participants in a botnet are referred to as "zombies" or "bots," and are controlled by a limited number of people and systems. Botnet architecture has evolved and scaled well over time. Early botnets used a traditional client-server hierarchy with commands sent in plaintext via the simple internet relay chat (IRC) protocol, and stopping a botnet was as simple as blocking incoming IRC traffic on one's network, and the botnet controller's IRC chat room was usually easy to find through simple analysis of a packet capture from the victim's network. Today's botnets have achieved a new level of sophistication and larger scale through the use of peer-to-peer architecture to control the zombies and obscure the controller, and

public key cryptography to digitally sign exploit tools and commands as in the case of the Gameover Zeus malware distributed via the Cutwail spam botnet [68]. Modern botnets are notoriously difficult to shut down, and continue to be a subject of great interest to the cybersecurity research community.

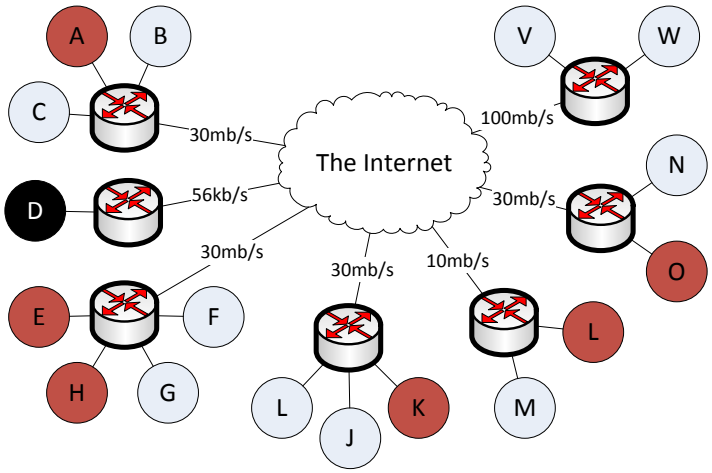

Figure 2.3 Node D has compromised nodes A, E, H, K, L, and O to form a botnet. D can order the botnet to attack V with a combined attack bandwidth of 130mb/s, overwhelming V's 100mb/s connection and denying legitimate access to services hosted by both V and its neighbor W. Neighbors of botnet nodes may also be impacted.

2.9.2 Hacktivism and Voluntary Botnets

Not all distributed attacks are performed by unknowing participant systems. In some instances these attacks are performed by users seeking to gain notoriety or create awareness of an ideological or political issue in what is known as hacking-based activism or "hacktivism." "Hacktivists" with minimal technical skills may install software developed by more skilled hacktivists to participate in a distributed attack either directly by initiating an attack in concert with other hacktivists. In the case of the popular and open source Low Orbit Ion Cannon (LOIC) software's [46] "hive mind" mode, hacktivists volunteer to join a botnet controlled by an IRC channel or RSS newsfeed and leave the software running unobtrusively on their computers without manual intervention. Disregarding past web browsing behavior such as searching for the LOIC software installer, using hive mind LOIC ostensibly gives the hacktivist a small degree of

plausible deniability: the hacktivist may deny being a willing participant and plausibly claim instead to be the victim of a botnet infection. Unfortunately for some participants, while the LOIC software has proven to be effective, it makes no attempt to disguise the hacktivist's source IP address. In just one notable example, more than a dozen hacktivists and LOIC users were identified and prosecuted for their participation in "Operation Payback" [58]. Operation Payback was a hacktivist campaign in late 2010 - early 2011 to launch DDoS attacks against various banks and payment processing services, purportedly in response to their refusal to process donations to WikiLeaks after WikiLeaks published a large collection of secret diplomatic messages from United States embassies and consulates to the State Department.

2.9.3 Defending Against Distributed Attacks

Interesting research topics in collaborative defense methods to reduce the power of DDoS attacks include path marking, where routers cooperate to identify sources of attack traffic, allowing the victim to distinguish attack traffic from legitimate traffic; and pushback, wherein routers cooperate to block the attack traffic as close to the attack source as possible [47]. DefCOM [39], developed at UCLA's Laboratory for Advanced Systems Research, is a promising collaborative defense system with which our CODON framework shares similarities. Unlike some collaborate defense systems that require all networks between the attacker and victim to participate, DefCOM provides DDoS defense using heterogeneous participants performing different tasks. DefCOM's defense functions are attack detection, traffic rate limiting, and traffic differentiation. Defense-related communication is performed over a peer-to-peer overlay network. DefCOM's research team claims DefCOM is effective with non-participants between the attacker and victim, and that the small scale tests they performed with real computers (around 200 nodes) is more realistic than other tests due to the infidelity of network simulation

software under DDoS conditions. While DefCOM shares some similarities to our work, namely the use of heterogeneous participants and peer-to-peer communication, our CODON framework takes a more holistic approach to security and threat information sharing rather than focusing solely on DDoS defense.

Many organizations subscribe to content distribution networks (CDN) such as Akamai [2] to more quickly deliver web pages and multimedia content instead of relying solely on servers on their own premises. These CDNs consist of servers at well-connected locations throughout the world where they can serve requests from global clients much more quickly than from a single location. The strategic positioning and high bandwidth of CDNs enables them to also work as load balancing systems for their subscribers, diminishing the effect of DDoS attacks by being able to sustain high traffic volumes and limiting the geographical regions of legitimate end-users that may be prevented from accessing CDN subscribers' systems by a successful DDoS. Comparing a DDoS attack to a large mob of people trying to enter the front entrance to an amusement park, using a CDN simply makes the entrance wider or adds more entrances to accommodate a larger mob. This is an effective strategy for network bandwidth-based attacks but is not cost effective for all organizations and will eventually fail to scale as attackers devise new bandwidth-intensive DDoS attacks. The CloudFlare [2] CDN also acts as a caching and filtering system to prevent malicious requests from reaching a subscriber's website. Research into methods to quickly deliver content from CDNs to end-users is ongoing and includes closer collaboration with ISPs [25], but we do not believe faster content delivery alone will necessarily contribute to security for the internet community as a whole.

Locasto *et al.*'s Worminator [37] project is a peer-to-peer alert distribution system that creates and distributes alerts from multiple organizations to generate inter-organizational threat

"watchlists" and seeks to provide a more global view of attacks with similarities to our CODON framework. While Worminator's goal of collaborative defense is similar, CODON differs greatly in the scale of participation. Whereas in [37] Worminator searches for suspicious traffic using homogenous IDS and alerting systems at four different sites within the United States, with the majority of alerts coming from the Columbia University's computer science department's network, we seek to leverage heterogeneous data sources and detection mechanisms based not only on dedicated IDS equipment in well-resourced organizations, but on a multitude of hosts including those of end-users worldwide.

2.10 Reputation-based Access Control: Blacklists and Whitelists

A very common method for controlling access to a limited or sensitive resource is through the use of lists known as blacklists and whitelists. Blacklists, sometimes also referred to as blocklists, explicitly enumerate who may not access a resource and whitelists conversely enumerate who may access a resource. Many systems are capable of using blacklists and whitelists to control access to network-based resources including firewalls, email servers, web servers, and remote systems administration servers. Blacklists are typically used to deny access to known offenders or abusers of a system. While a whitelist serves as the authoritative listing of who may access a resource, additional actions may be required before that resource may be accessed. In the context of email servers, blacklists known as DNS-based blackhole lists (DNSBL) [66] are used to ignore or refuse service requests from systems that have earned a reputation for sending undesirable email such as spam or malware. Some individual email users employ a whitelist to only receive email messages from senders they know and trust, thereby protecting them from unknown senders.

Blacklists and whitelists can be very effective access controls for protecting a resource, but they often require substantial effort to accurately maintain and keep up to date over time as system requirements change and the internet grows. The very restrictive nature of whitelisting tends to limit its usefulness to special cases and small-scale/intra-network use since all the potential users of a resource must be known in advance. In contrast to whitelisting, both the burdens and benefits of blacklist maintenance can be shared by many organizations and people. Collaborative blacklist maintenance and distribution allows a large number of systems to benefit from the negative experience of relatively few victims. Therefore blacklisting is commonly employed by email servers and firewalls for protection from notorious malicious systems.

False positives will inevitably enter collaboratively maintained blacklists. These false positives could be the result of mistakes by blacklist contributors, but are more commonly caused by transient misbehavior by a normally benign system. This transient misbehavior may be due to improper system configuration, system compromise by an attacker, or the misbehavior of a limited number of system users inside a small area of a larger network. It can be difficult to convince blacklist maintainers that a false positive is in fact a false positive, even if a blacklist maintainer uses dedicated staff to handle removal requests as in the case of the popular Spamhaus [66] and SORBS [54]. Because different collaboratively maintained blacklists may use different criteria to judge whether a system is malicious, it may be beneficial to use multiple blacklists to restrict access to a system. Blacklist lookup tools such as the Anti-Abuse Project's Multi-RBL Check [4] are available to aid in determining which common blacklists include a particular system. These tools can be useful to system administrators investigating suspicious activity observed on their systems by the administrators to correlate the suspicious host's source IP address with the observations of multiple blacklist maintainers. Blacklist lookup

tools are also useful to the administrators of false positive or formerly compromised systems as it can be difficult to know which particular blacklist(s) may be causing other systems to block legitimate communication.

The OpenBL.org project [49] maintains blacklists based on sources of attacks detected by almost four dozen servers distributed across the globe. When software on one of these servers detects brute force login attempts or certain other webserver-related attacks, the software automatically emails the attacking system's presumed owner based on contact information gleaned from the Whois databases of regional internet registries and the DNS SOA (Source of Authority) resource record corresponding to the attacker's IP address. The project managers publish inception-to-date historical attack statistics as well as blacklists that are based on specific attacks and the past 90, 60, and 30 days' data. Unfortunately, despite the "open" in its name, only the blacklists themselves and some statistics are available the general public to access. The public can only contribute to the project by donating servers or money, not by running OpenBL's software on their own systems or directly contributing to software development and system management.

2.11 Collaborative and Cooperative Defense

The power and efficacy of a distributed attack largely depends on the amount of resources at the attacker's disposal being greater than the amount of resources at the victim's disposal. Increasing the victim's resources or efficiency of resource use is a common way to defend against a distributed attack, but this scaling of resources to correspond to a scaling attack can be quite costly. As an alternative defensive approach that reduces the need for expensive resource scaling, the victim can collaborate with other systems to slow down and possibly stop the attack's resource usage. In some situations, a simple phone call or email to the

administrator of a zombie-infested network may be sufficient to disconnect infected hosts from the network and cripple the attacker's resources. However, administrator-to-administrator communication is not as effective in a widely-distributed attack and can be hindered by the usual human communication barriers of language, time zone differences, administrator availability, organizational priorities, and even indifference to the victim's plight.

In November of 1988, Cornell University student Robert Tappan Morris unleashed the so-called Morris Worm [65], allegedly in an attempt to measure the size of the internet. The worm exploited common vulnerabilities in UNIX software and resulted in an unprecedented self-perpetuating DoS attack as the victim computers became infected by the worm multiple times and slowed down further with each infection. In response to the Morris Worm outbreak, the first computer security incident response team (CSIRT), known as the CERT Coordination Center or simply CERT/CC [63], was formed at Carnegie Mellon University's Software Engineering Institute. The goal of a CSIRT is, as the name implies, to provide a quick and coordinated response to security incidents from detection to remediation and taking steps to prevent similar incidents in the future. A CSIRT is often activated on an as-needed basis and consists of a subset of key staff in an organization with complementary roles such as security analysts, firewall administrators, system administrators, and managers. Some larger CSIRTs such as CERT/CC and the United States Computer Emergency Readiness Team (US-CERT) serve broader segments of the internet community on an ongoing basis by publishing guidance about cybersecurity best practices and disseminating announcements about important software updates, and high-profile threats and attacks.

Information sharing and analysis centers, or ISACs, are coalitions of critical infrastructure professionals formed to facilitate sharing and analyzing information about threats to common

interests in specific infrastructure sectors. ISACs exist for many sectors including the research and higher education community (REN-ISAC) [57], and the threats in which they are interested are not strictly computer based as in the case of Supply Chain ISAC [59]. Since the information shared within an ISAC is based on the experience of fellow community members and the analysis of this information is similarly focused toward benefiting the same community's security, an ISAC can be a very valuable and relevant source of security information. The sensitivity of the threat information that is sometimes shared within an ISAC often requires a moderate to high level of trust to be established with a participant before the participant may receive deep or meaningful analysis of threat information from the ISAC, sometimes leaving marginal participants and the public completely unaware of useful information generated through the ISAC's efforts.

Larger CSIRTs such as US-CERT and ISACs commonly use "newsfeed" technologies like Really Simple Syndication (RSS) and email discussion lists to share security-related announcements, but these announcements typically take the form of verbal descriptions in a loosely structured format. Unlike antimalware or IDPS signature subscriptions, these announcements are intended for human consumption and place the burden on interested system administrators to read and manually act on the announcements. CSIRTs and ISACs form a valuable segment of the internet community, and we wish to incorporate their efforts into a scalable and automated cooperative defensive system that even the general public may use to level the figurative playing field between relatively small victims and ever-growing and adaptive teams of attackers.

2.12 Crowdsourcing and Wikis

"Crowdsourcing" is a term used to describe the use of a large and often geographically distributed number of people with varying skill levels to perform a large task. Perhaps the most well-known and successful example of crowdsourcing is the much-acclaimed Wikipedia online encyclopedia, which at this time hosts more than 31.9 million pages and bills itself as "the free encyclopedia that anyone can edit" [71]. Collaborative web publishing systems and the general model upon which Wikipedia and others are based are commonly referred to as "wikis" [15].

Crowdsourcing has proven to be very effective for accomplishing large tasks like documenting a wide breadth of topics online, but it is not without its pitfalls; organizational and policy issues exist since anyone can modify the system's content. Wiki content is often "vandalized" to promote an ideology either contrary or completely unrelated to the wiki's goal, or to shock, disgust, or amuse large numbers of wiki viewers. The quality of the wiki's information can also be diluted by libelous information contributed by users with subversive or malicious intent, or even through simple ignorance, and contributors introduce or perpetuate bias that can be difficult to control without dedicated editorial oversight by other users. Common safeguards that have been developed to protect wikis against low-quality contributions include requiring all contributors to create a user account to log and track their historical activity, and the use of domain experts and reviewers with a history of positive contributions to review and approve wiki content changes. Employing more safeguards generally raises the bar for participation and discourages casual users from contributing information, which in turn may make the collaborative system less useful and further discourage contributions due to lack of popularity. Despite the use of safeguards against vandalism, it is still possible to spread misinformation through a wiki. Mainstream news outlets with editorial staff

have relied on Wikipedia articles containing fictitious information, leading to public embarrassment as in the case of [26].

2.13 Crowdsourced Threat Identification via Web Browser Add-ons

Tools such as SmartNotes [60] rely on end-user participation via a web browser add-on to actively mark websites as being malicious or untrustworthy, and machine learning systems can be used to analyze such dynamically-produced data [24]. Prominent web browser tools such as AdBlock Plus [23] automatically block the downloading of obtrusive advertisements and optionally malware through the use of so-called community-maintained "filterlists." Filterlists are community-maintained pattern-based blacklists. In the case of the popular EasyList filterlist [19], changes are suggested by end-users in community discussion forums and later approved and distributed by community project leaders. While the CODON framework focuses on automated collection and distribution of threat sources, web browser based tools could trivially be integrated with CODON through a simple menu option, e.g. “report this page as malicious,” that logs the pertinent details for consumption by the CODON Sensor software, as we will discuss in Section 3.3.3.

2.14 Collateral Defense and the Neighborhood Watch Model

Two nontechnical examples that perhaps best summarize the concept of collateral defense are neighborhood watch programs and public notices. In a neighborhood watch program, residents of a neighborhood typically receive basic safety awareness training and agree to monitor the neighborhood for suspicious activity during their day-to-day activities and report such activity to local law enforcement personnel as well as fellow residents. Public notices are commonly posted to community billboards at grocery stores, on or near community

mailbox clusters, and at post offices to disseminate information about recent criminal activity, missing children, and upcoming events. In both of these nontechnical examples, the goal is to leverage the abundance of potential observers of criminal activity to assist the relatively few enforcers who are specially trained and equipped to directly address the perpetrators. Due to the ubiquity of consumer internet access and low level of technical skill required to connect to the internet, the vast majority of internet users lack the technical, let alone cybersecurity-specific, resources available to large and well-funded organizations. We have designed the CODON framework to better leverage the disparity between "observers" and "enforcers" on the internet in the interest of protecting the internet community as a whole. Whereas distributed attacks such as a DDoS can cause collateral damage to a victim's neighbors, CODONs seek to provide collateral defense to neighbors of participants.

Our CODON framework describes a peer-to-peer overlay network formed either ad-hoc, which is the default and preferred method, or by formal agreement between two or more participants for the purpose of contributing to one another's collective health and defense through timely and automated security-related information sharing. A geographically unbalanced CODON can provide value to its participants and neighbors within a geographical region, and a geographically balanced CODON can have an even greater impact by benefiting its many non-participant neighbors in many regions. Participants in a CODON may include AS administrators, research institutions, small corporate networks, computer enthusiasts, and novice home users. Indeed, a CODON will be most effective when comprised of a multitude of heterogeneous participants.

We will describe the CODON framework in depth in Chapter 3, but Figure 3.1 provides a high level overview. The scope, level of detail, and accuracy of information provided by a

CODON participant will vary based on that participant's capabilities. Some participants operate web servers and networks that are well-suited for publishing relatively static content like software updates or more dynamic content like blacklists with a low update frequency. Casual but technically adept internet users may work in loosely-knit teams to review and refine blacklists in their free time. Private sector and academic research labs may participate in sensor projects like the UCSD Network Telescope [14] to collect information about large-scale probing activity, or in some cases even track and seize control of botnets. Smaller organizations and consultants may have limited computing resources but can assess other participants' network perimeter security via regular or ad-hoc vulnerability scans. Other organizations are uniquely placed as transit providers or ISPs and can perform bandwidth throttling or blocking of malicious traffic in response to a distributed attack, or detect large scale attacks that cannot be detected based only on sensor data from end-users' computers.

3 CODON Architecture

A core design goal of the CODON framework is the encouragement of automated sharing and usage of information about malicious activity. Each CODON participant fills at least one role in the CODON based on the resources the participant desires to contribute. These roles, summarized in Figure 3.1 and Table 3.2, focus on sharing security incident information and coordinating defensive activity. To maintain transparency and trust among CODON participants, and to promote the broadest base of adoption possible, we strongly urge participants to use open source software whenever possible for CODON operations. For example, peer-to-peer communication systems such as Chord [67], GNUnet [31] or Trust-X [22] should be used for organizing the various participants into an overlay network and OpenSSL [50] should be used to implement the various cryptographic functions required. Established and emerging open standards are used throughout our framework in the spirit of honesty and to encourage participation on an international scale. Options affecting privacy, performance, and network traffic volume should be exposed to the user via administrative utilities designed to provide a familiar user experience specific to the operating system on which the CODON software is installed. For example, in Linux the software configuration should be exposed through popular window managers' settings menus and configuration files should be stored in `/etc`, `/usr/local/etc`, or another directory consistent with the particular Linux distribution, whereas in Apple OS X the configuration tools should function and appear similarly to other "System Preferences" panes.

As we discussed earlier in Section 2.12, there is naturally great potential for abuse in systems that rely on a massive number of loosely affiliated users providing input for public consumption. For this reason, we have taken care to incorporate safeguards against abuse in

our design of the CODON framework. Encryption is used to reduce the likelihood of message tampering and selective message blocking by malicious third parties. A small number of reports of malicious activity only results in a small penalty for the offending entity. Misbehaving participants are silently ignored by other participants via an exponential back-off algorithm, CODON-wide informational updates are distributed peer-to-peer, and session state is kept to a minimum to reduce the havoc caused by a misbehaving participant. As a CODON grows in size, the effect of mischievous participants should be drowned out by the volume of good information provided by normal participants. All date and time data in the CODON are communicated in Coordinated Universal Time (UTC) to aid in correlating reports of malicious activity.

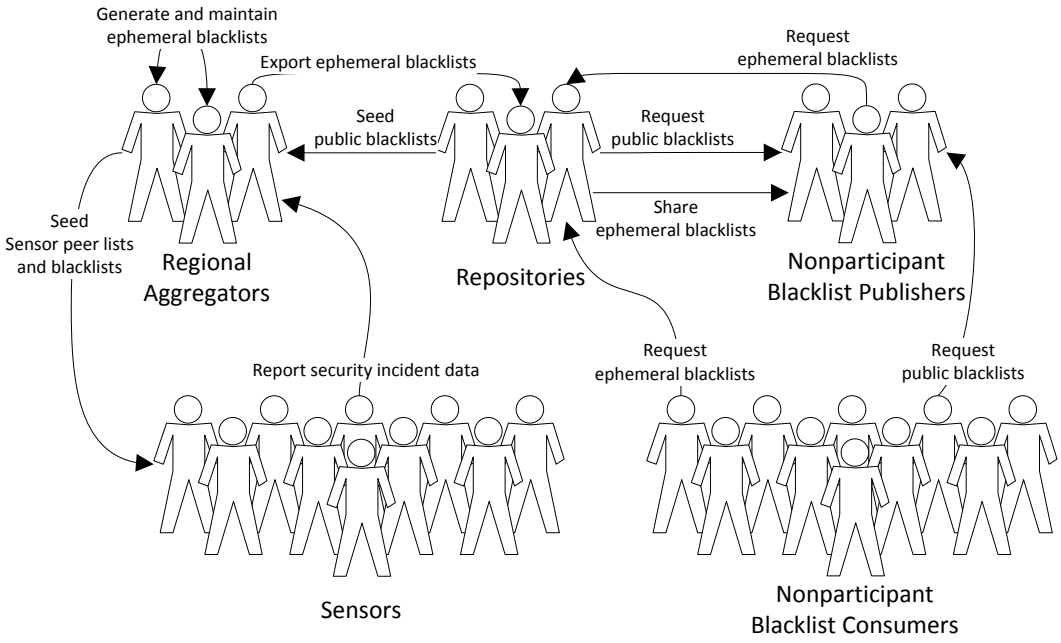

Figure 3.1 A high level overview of security information sharing in a CODON. Even non-participants may benefit from the CODON's efforts.

3.1 Ephemeral Blacklists

We will now describe ephemeral blacklists, a key concept in the CODON framework, in greater detail. An ephemeral blacklist is a blacklist containing IP addresses and subnets that

have been reported by CODON Sensors to have engaged in malicious or suspicious activity, each with a corresponding expiration timer. Ephemeral blacklists are compressed using common compression algorithms before transmission over the network whenever feasible to reduce the volume of network traffic sent by CODON participants. Consider our example of a very small ephemeral blacklist in Table 3.1. Each ephemeral blacklist is marked with a Type ID, an arbitrary but consistent 4 byte positive integer value corresponding to the type of information contained in the ephemeral blacklist. For example, a Type ID of 0x0001 may correspond to mail servers being used as spam relays, and 0x1336 may correspond to hosts caught attempting to repeatedly and rapidly guess user credentials via a web-based login form. When a Regional Aggregator receives a report of malicious activity via an Information Sharing Message (ISM, described in Section 3.4), it updates the appropriate ephemeral blacklist based on the Type ID included in the ISM.

Suspected malicious hosts are temporarily blacklisted based on two criteria: whether a report is the first reported offense by a particular host, and the number of unique Sensors reporting the activity. If a suspected malicious host does not yet exist in the ephemeral blacklist, its IP address is temporarily stored in a “watchlist” for the current and next update cycle. Update cycles are explained in more detail throughout our discussion of Regional Aggregators in Section 3.3.1. If the reported host is already in the watchlist, it is moved to the ephemeral blacklist and given a short expiration of 15 minutes. However, if the reported host is already in the blacklist, its blacklist expiration increases. Each Sensor may only cause a particular expiration to increase once every two update cycles, effectively preventing a single host from blacklisting another by requiring multiple witnesses to report the same observation before it is acted upon.

Table 3.1 Sample ephemeral blacklist of hosts engaging in port scanning activity

(Metadata): Ephemeral Blacklist Type ID 0x0002, "Port scanning, 100 ports in < 5 seconds"	
Offending IP or CIDR subnet	**Expiration (UTC)**
10.0.0.1	2014-01-01 14:00
10.1.0.0/24	2014-01-02 13:00
10.3.3.7	2014-01-01 12:15
10.50.2.0/31	2014-01-02 12:35

As more hosts are added to an ephemeral blacklist over time, the blacklist is condensed when possible by grouping multiple IPs together into a single entry using classless inter-domain routing (CIDR) notation. CIDR notation is a concise means of denoting subnets using an IP address and a mask known as a "prefix". The IP address range 10.0.0.0 - 10.0.0.255 is represented in CIDR notation as 10.0.0.0/24, where 10.0.0.0 is the beginning of the IP address range and /24 is the prefix, meaning that the first 24 bits in the given IP address remain the same and only the final 8 bits (the final octet of the IP address) can vary. Similarly, 10.0.0.0/16 represents the IP address range 10.0.0.0 - 10.0.255.255, and 10.0.0.0/32 represents the single IP address 10.0.0.0. We only condense multiple IPs into CIDR notation when the consecutive IP addresses have expirations of twelve or more hours, causing the multiple consecutive IPs to be treated as a single entity using the largest expiration that applied to any of the consecutive IPs. Furthermore, once more than half of a /24 subnet is blacklisted, we treat the entire /24 subnet as malicious. We believe this is reasonable behavior based on Shue *et al.*'s observations about predominantly malicious ASes in [61]. Non-participants wishing to make use of ephemeral blacklist data published by CODON Repositories may be most interested in blacklisting hosts and subnets with expirations of at least 24 hours.

3.2 Contributions by and to CODON Non-participants

As we have already discussed, the CODON framework is designed to benefit CODON participants and non-participants alike. Likewise, a CODON certainly benefits from the efforts of non-participants. The existence of a CODON would clearly be impossible without the efforts of the many non-participants who contribute to the infrastructure of the internet as a whole. We have grouped the key non-participants that are functionally most aligned with the CODON into two categories: Security Information Aggregators and Remediators.

3.2.1 Security Information Aggregators

Security Information Aggregators are non-participants who collect and curate, whether manually or with automated tools, security-related information that may be beneficial to the public. Examples of such information include lists of software vulnerabilities, IP ranges and DNS hostnames known to be used predominantly by malicious actors, malware distribution URLs, and hashes of dangerous files. Many organizations and individuals subscribe to Security Information Aggregators' services, sometimes paying a fee or donating money in return for data that can be manually or automatically imported into their systems. For example, antimalware software vendors commonly charge a subscription fee for the ability to download new malware detection signatures, although they sometimes allow their antimalware software to continue functioning with an old signature set if the customer does not renew the subscription. Google provides a free service called Safe Browsing [30] and a corresponding API to query a list of potentially malicious or compromised websites, and the service is incorporated into Mozilla's popular Firefox web browser and Google's own Chrome browser. CODON leverages the "free" work of Security Information Aggregators to protect participants, and it is therefore only

appropriate that the CODON contribute as much relevant actionable intelligence back to these Security Information Aggregators as possible.

3.2.2 Remediators

Remediators are organizations that volunteer or are paid to fix network health and security problems. Application-level examples include software vulnerability patch creation and publishing by software developers and/or hosting services. This encompasses commercial software vendors' update services, Linux package mirrors and repositories, popular open source software distributors like SourceForge [17] and GitHub [29], and security research organizations that publish temporary fixes for zero-day exploits. Network-level Remediators include those that either throttle or filter DDoS attack traffic to improve the quality of service for legitimate traffic flows as in the case of AT&T's "scrubbing" system described in [7]. Also included in the Remediator category are CDNs that specialize in load-balancing traffic to protect their customers from DDoS attacks and flash crowds.

3.3 CODON Participant Roles

Participants in a CODON serve in one or more roles, as we will explain in more detail below. All CODON participants serve in the Sensor role by default but are encouraged to serve in additional roles as their resources may permit. Table 3.2 summarizes each role in the CODON framework, the role's ease of participation or "minimum buy-in," and examples of the various organizations and people we believe are likely to participate in each role.

Table 3.2 CODON participant roles

Role	Ease of Participation	Likely Participants
Sensor	Easy to difficult; requires the installation of a software service and access to minimal bandwidth. Active detection can be done with free software and low bandwidth. Large scale passive detection may require access to traffic monitoring tools or specialized hardware.	Novice computer users, advanced/enthusiast computer users, universities, corporations, governments, researchers, information security organizations internet service and transit providers, web hosting and cloud computing providers.
Regional Aggregator	Medium; requires some database storage to manage participant state and blacklist data, and bandwidth to accommodate join/leave requests from many participants and to correlate and share incident data with other Regional Aggregators.	Advanced/enthusiast computer users, universities, corporations, researchers, internet service providers, regional CSIRTS and ISACs, cloud computing providers.
Repository	Medium to difficult; requires a web server with good bandwidth and storage, ability to receive and share data from Regional Aggregators and non-participant Security Information Aggregators, e.g. spam blacklists, botnet zombie lists.	Internet service providers, universities, software development companies, regional and industrial CSIRTs and ISACs, web hosting providers.
Defensive Service Broker	Easy to medium; requires a web server with reasonable uptime, bandwidth, and processing resources for effective matchmaking.	Small companies, universities, information security organizations, non-profits.

3.3.1 Regional Aggregator

Regional Aggregators are, as the role name implies, responsible for aggregating security and threat information reported by CODON participants. Aggregating reports from thousands of participants and working in concert with other Regional Aggregators to provide the CODON with a regular global view of threats is a resource intensive task best suited to participants with reliable internet connectivity, modest bandwidth, midrange computing power, and system administrators with a better technical understanding than novice end-users. Most CODON participants will only fill the Sensor role, and the CODON Sensor software should have minimal resource requirements in order to encourage adoption and acceptance by users with limited technical knowledge. As mobile computing continues to proliferate in the form of lightweight laptops, smartphones, and other increasingly powerful portable devices, there is a clear need

for relatively well-resourced and well-connected participants to provide the fluctuating swarm of participants with stability and coordination. In this sense, Regional Aggregators are similar to Skype supernodes and BitTorrent trackers as discussed in Section 2.2; but unlike the Skype model, we do not believe it is necessary for a CODON's Regional Aggregators to fall under the management of a single entity to be effective. In Figure 3.2, we illustrate the process by which a Regional Aggregator updates an ephemeral blacklist.

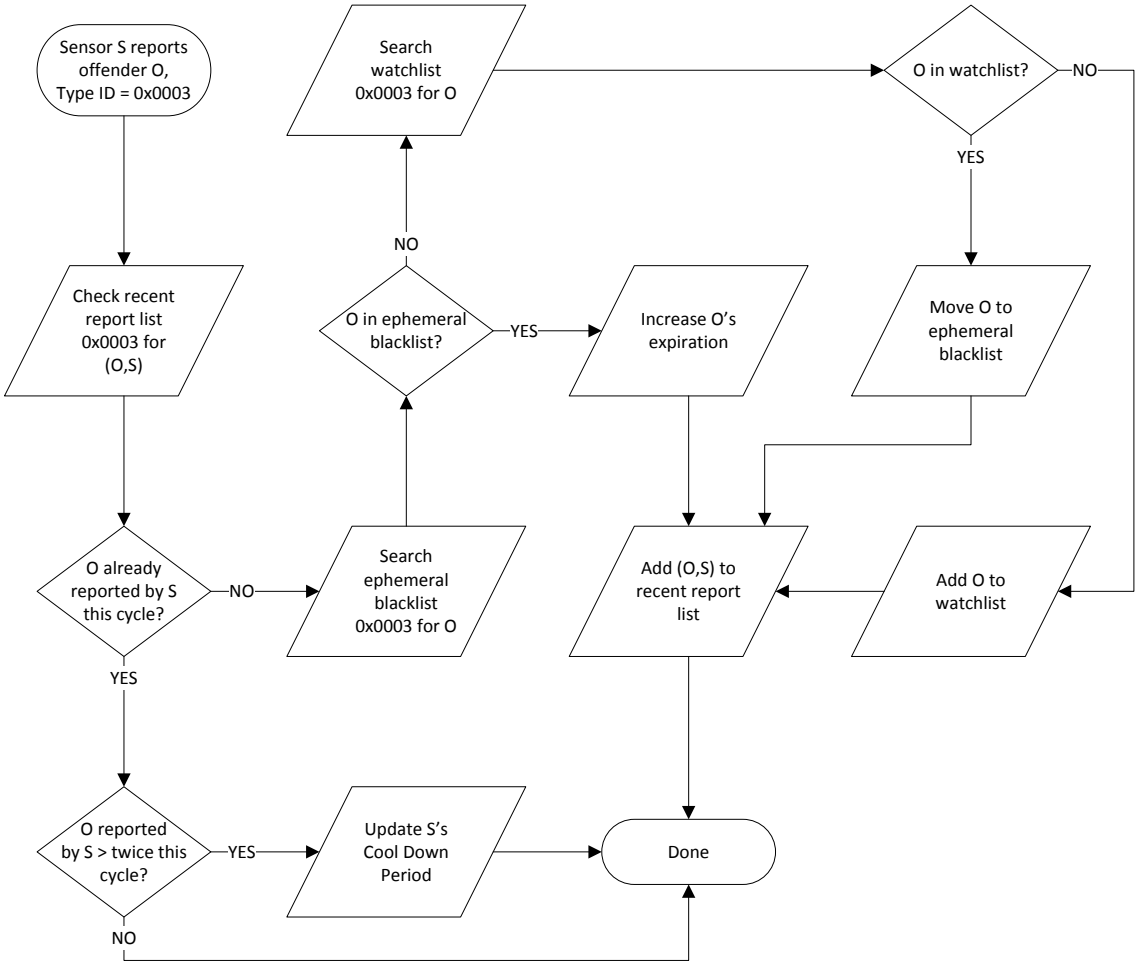

Figure 3.2 A Regional Aggregator receives an ISM from Sensor S reporting that offender O is distributing malware. The Regional Aggregator considers updating its corresponding ephemeral blacklist. If not subscribed to the ISM's Type ID, the Regional Aggregator passes the ISM to subscribing Regional Aggregators via the overlay network.

3.3.1.1 Participant State Database

A participant may wish to limit the resources consumed by the system on which the CODON Regional Aggregator software operates. When the resource limit is reached, the Regional Aggregator should temporarily reject join attempts by new participants until sufficient resources are available to accommodate additional participants. Reasonable resource thresholds include a maximum number of concurrent participants, a maximum size for the Participant State Database, and a limit to the Regional Aggregator software's mean network or CPU utilization. Resource usage can be further limited by only subscribing to a subset of Type IDs, but even modestly equipped modern computers with multithreading and sufficient RAM in conjunction with free database software should prove quite capable. Table 3.3 lists the state data a Regional Aggregator must manage for each participant. Notwithstanding overhead incurred by the particular database software employed to store and update participant state data, approximately 16.5 million participants' 65-byte state could be stored in a 1 GB database, and we expect other resource-intensive tasks such as security incident data correlation, ephemeral blacklist management, and communication encryption overhead to be the primary causes of resource utilization for Regional Aggregators.

Table 3.3 Participant state data maintained by a Regional Aggregator.

Participant ID (UUID), 16 bytes	**Participant Type Mask, 1 byte**	**WAN IP, 4 bytes**	**Participant Time Offset, 8 bytes**	**Age, 2 bytes**	**Cool Down Period, 2 bytes**	**Session Key, 32 bytes**
79f91125-0118-4f38-9c72-ed1bb41fc798	0	10.1.0.1	-401000000000 (-401 sec.)	20 (1hr., 40min.)	16 (1hr., 20min.)	256 bit session key agreed upon during `participant_join`
36dc3aab-397d-46d8-8c05-b506f890d910	1	10.2.0.3	88200000000 (+88.2 sec.)	600 (2d., 2hrs.)	0	256 bit session key agreed upon during `participant_join`
3bc27d1c-5bd5-4bd8-a367-acdcb435aa13	2	10.5.2.1	15800000000 (+15.8 sec.)	808 (2d., 19hrs., 20min.)	0	256 bit session key agreed upon during `participant_join`

The Participant UUID is an RFC 4122 compliant 128-bit version 1 universally unique identifier (UUID) [36]. The timestamp portion of the UUID is based on the beginning of the participant's session establishment process from the Regional Aggregator's perspective. RFC 4122 requires that the timestamp be based on the UTC representation of 100 nansecond intervals elapsed since 00:00:00.00, 15 October 1582.

Participant Type Mask is a bit mask indicating which roles in addition to the mandatory Sensor role are filled by the participant. A value of 0 denotes that the participant is a Sensor, 1 indicates the participant is a Repository, 2 indicates the participant is a Regional Aggregator, and 4 indicates the participant is a Defensive Service Broker. Other powers of two are reserved for future use. Participants serving in multiple roles have a Participant Type Mask value equal to the sum of their individual roles. For example, a Participant Type Mask value of 3 indicates that the participant is a Sensor, a Repository, and a Regional Aggregator (0 + 1 + 2).

The Participant Time Offset is essential for the Regional Aggregator to properly calculate ephemeral blacklist updates and correlate security incident data reported by multiple participants. This offset is stored as an 8 byte value representing the difference between the Participant's reported time and the Regional Aggregator's system time in 100 nanosecond intervals at the beginning of session establishment. The calculated time offset does not account for communication latency between the participant to the Regional Aggregator, but this latency may be disregarded for our purposes due to our use of update windows as described later in this section. Only 35 bits are necessary to store the Participant Time Offset because we do not permit the Participant's clock to differ from the Regional Aggregator's clock by more than ±900 seconds (15 minutes), which is 18,000,000,000 100 ns intervals and can be stored in 35 bits using the two's complement representation. While we could have chosen a 5 byte field size, 35

bits is 4.5 bytes, and in the interest of simplified database storage and computation, we chose to round up to the next power of two, which is a more easily managed 8 bytes. Participants are temporarily assigned a placeholder Partipant ID of 00000000-0000-0000-0000-000000000000 until the `participant_join` process, described in Section 3.3.1.3, is complete and the session with the Regional Aggregator has been established.

Cool Down Period indicates the remaining number of update cycles the Regional Aggregator should ignore updates provided by the participant due to misbehavior or other difficulty communicating with the participant. An update cycle occurs every five minutes, and thus a Cool Down Period of 288 indicates that the Regional Aggregator should ignore the participant for one day (1,440 minutes). Each update cycle in which the participant has misbehaved results in the cool down period doubling via a simple truncated exponential back off algorithm with a maximum permitted Cool Down Period of 512 (1 day, 18 hours, 40 minutes). Each update cycle in which the participant has not misbehaved results in the Cool Down Period decrementing by one until the Cool Down Period eventually reaches zero.

The session age, in update cycles, is stored in the Age field. Sensors have a maximum age of 512 update cycles (1 days, 18 hours, 40 minutes), and participants serving in additional roles have double that maximum age. At the end of each update cycle, the Age field is incremented until the maximum age is reached. Once the maximum Age is reached, the Regional Aggregator sends a `participant_rejoin` message to the participant and removes the participant from its Participant State Database. This process helps ensure that inactive participants that were unable to complete the `participant_leave` process are eventually culled from the CODON to reduce unnecessary resource consumption while avoiding the need for regular “heartbeat” traffic between participants and Regional Aggregators. A secondary

effect of the aging process is that it redistributes participants among Regional Aggregators to rebalance their loads, although this load balancing process is admittedly naïve.

The WAN IP is simply the WAN IPv4 address from which the participant successfully connected to the Regional Aggregator. Although we track the WAN IP address of participants, the Participant ID and Session Key are the primary fields used to identify registered participants. It is possible for multiple participants to use the same WAN IP address, as may be the case in a large organization containing multiple participants that use PAT or NAT to access resources outside the organization's network. When a Regional Aggregator experiences misbehavior from a WAN IP address whose communication is not encrypted with a Partipant ID's corresponding Session Key, it is assumed that the misbehavior is being caused by a non-participant. Such misbehavior may take the form of a SYN flood or similar misbehavior such as the host beginning a TCP 3-way handshake with the Regional Aggregator by sending a SYN and failing to respond to the Regional Aggregator's SYN-ACK. An important benefit of this approach is that participants with active connections to the Regional Aggregator are not punished by misbehavior from hosts spoofing their source IP address.

The Session Key is a 256 bit symmetric encryption key generated by the participant and agreed upon by both the participant and Regional Aggregator during session establishment. The session establishment and key generation processes are described in more detail below in Section 3.3.1.3 and in our discussion of Sensors below in Section 3.3.3. The session key is used by the Regional Aggregator to encrypt communications sent to the participant such as ephemeral blacklist updates, and from the participant to encrypt communications such as security incident data sent to the Regional Aggregator. Despite the use of cryptography, we are not particularly interested in preventing eavesdroppers from learning the content of messages

sent from the Regional Aggregator to participants. Rather, we use cryptography to reduce the risk of a third party intercepting and tampering with these messages, possibly causing participants to block hosts that were not originally included in ephemeral blacklist updates from the Regional Aggregator. Encrypting communications to the Regional Aggregator also reduces the risk that a malicious host would join the CODON and re-transmit information recently sent by another participant to more quickly or heavily penalize hosts reported by the innocent participant.

3.3.1.2 Communication Among Regional Aggregators and Repositories

Regional Aggregators must communicate with one another regularly to share security incident information and ephemeral blacklist data generated during each update cycle. An update cycle occurs every five minutes, and Regional Aggregators send an Activity Change Set detailing activity that occurred during the past update cycle to fellow Regional Aggregators using a peer-to-peer distributed hash table (DHT) algorithm such as Chord [67]. Activity Change Sets, of which we provide an example in Table 3.4, are marked with the same 4 byte Type ID that is used for managing ephemeral blacklists as already detailed in Section 3.1. Other maintenance tasks related to keeping security incident information up to date are performed at the end of each update cycle, such as expiring stale Sensor reporting history, updating participant Age and Cool Down Periods in the participant state database, and consolidating consecutive IP addresses into CIDR subnets in ephemeral blacklists. A Regional Aggregator may choose to “subscribe” to certain Type IDs and must use the information it receives from other Regional Aggregators to update its own ephemeral blacklists corresponding to those Type IDs. However, all Regional Aggregators must accept security information received from Sensors corresponding to other Type IDs and relay those messages to subscribing Regional Aggregators through the overlay

network. We believe it is neither feasible nor likely that all Regional Aggregators will always have a complete and consistent view of all security data shared in the CODON as the CODON scales due to transient issues like connectivity failures and churn among Regional Aggregators. Once per hour, after sharing Activity Change Sets, all Regional Aggregators must share and merge their ephemeral blacklists to reach a consistent understanding of ongoing malicious activity. This hourly update is then shared with Repositories for further distribution as discussed in Section 3.3.2.

Table 3.4 An example Activity Change Set distributed by a Regional Aggregator during shutdown

(Metadata): Activity Change Set Type ID 0x0002, "Port scanning, 100 ports in < 5 seconds"		
(Metadata): Source Regional Aggregator Participant ID 3bc27d1c-5bd5-4bd8-a367-acdcb435aa13		
Offender IP	**Offense last reported (UTC)**	**Offense count by unique Sensors**
10.1.0.8	2014-01-01 11:01:00	1
10.13.0.3	2014-01-01 11:02:15	2
10.33.7.57	2014-01-01 11:04:23	5

When a Regional Aggregator initializes, it generates a self-signed 4096 bit RSA key pair and publishes its own Participant ID, WAN IP, and public key to all Repositories that it can locate. The CODON software package should include an initial seed list of Repositories to allow Regional Aggregators to quickly enter service without the need to manually locate a Repository. Whenever a Repository requests ephemeral blacklist data from a Regional Aggregator, the Regional Aggregator will provide its most recently merged complete ephemeral blacklist corresponding to the Type ID(s) requested by the Repository. In addition to ephemeral blacklist data, the Regional Aggregator will provide the requesting Repository with a list of all its known peers in the Regional Aggregator overlay network including the corresponding Participant IDs, WAN IPs, and public keys. When a Regional Aggregator begins the process of shutting down, e.g. for scheduled system maintenance or because its maximum age has been reached, it will attempt to quickly distribute an Activity Change Set to its peers in the Regional Aggregator

overlay network before performing the usual `participant_leave` process described in Section 3.3.1.3. Regional Aggregators that have announced their departure to the overlay network or have been unresponsive for three consecutive update cycles are included in the peer list provided to Repositories but flagged for removal. Repositories may request an updated Regional Aggregator peer list only twice per hour to limit resource usage, and will use the peer list to aid in directing new participants to Regional Aggregators. If a Repository is unable to contact a Regional Aggregator in two consecutive attempts, it will consider the Regional Aggregator dead and select a new Regional Aggregator from which to request updates.

3.3.1.3 Participant Session Management

We will now discuss session management between Regional Aggregators and other participants. The three major session management processes, described in more detail below, are: `participant_join`, `participant_leave`, and `participant_rejoin`.

In the `participant_join` process illustrated below in Figure 3.3, if the participant is new to the CODON it locates a Regional Aggregator with whom to register by contacting a well-known Repository over HTTP. A partial list of Repositories should be included with the CODON software installation package to make this process as simple for the end user as possible. If the participant has previously participated in the CODON, it will attempt to connect to a Regional Aggregator included in its most recent seed list of Regional Aggregators, and if unsuccessful fall back to contacting a Repository. The Repository generates a seed list of up to five Regional Aggregators that appear to be near the participant, along with each of those Regional Aggregators' corresponding public RSA keys previously published to the Repository. These Regional Aggregators may be chosen based on an AS number lookup, IP-based geolocation, anycast [1] DNS address resolution, or another method the Repository may wish to employ, but

a reasonable effort should be made to avoid always providing the same one or two Regional Aggregators to the same participant.

Having received a list of Regional Aggregators with which it may register, the participant attempts to register itself with a Regional Aggregator by encrypting a `participant_join` request using the Regional Aggregator's public RSA key and sending the request to the Regional Aggregator. The `participant_join` request consists of a 256 bit Session Key randomly generated by the participant and the participant's current system time as described above in Section 3.3.1.1. The Regional Aggregator will accept any 256 bit Session Key except an empty/null key. If the participant is not accepted by the Regional Aggregator due to resource constraints, imminent shutdown, or unacceptable clock skew as mentioned in Section 3.3.1.1, the Regional Aggregator will reply with the plaintext ASCII string `CODON-NAK-BUSY`, `CODON-NAK-DOWN`, or `CODON-NAK-TIME`, respectively, causing the participant to restart the registration process with a new Regional Aggregator. If the participant is accepted by the Regional Aggregator, the Regional Aggregator will send an acknowledgement containing the participant's newly generated Participant ID and a seed list of participants from its participant state database to serve as peers in the participant overlay network, all encrypted using the Session Key. If the participant is rejected twice due to clock skew, the participant's CODON software should notify the end-user that the system time needs to be updated.

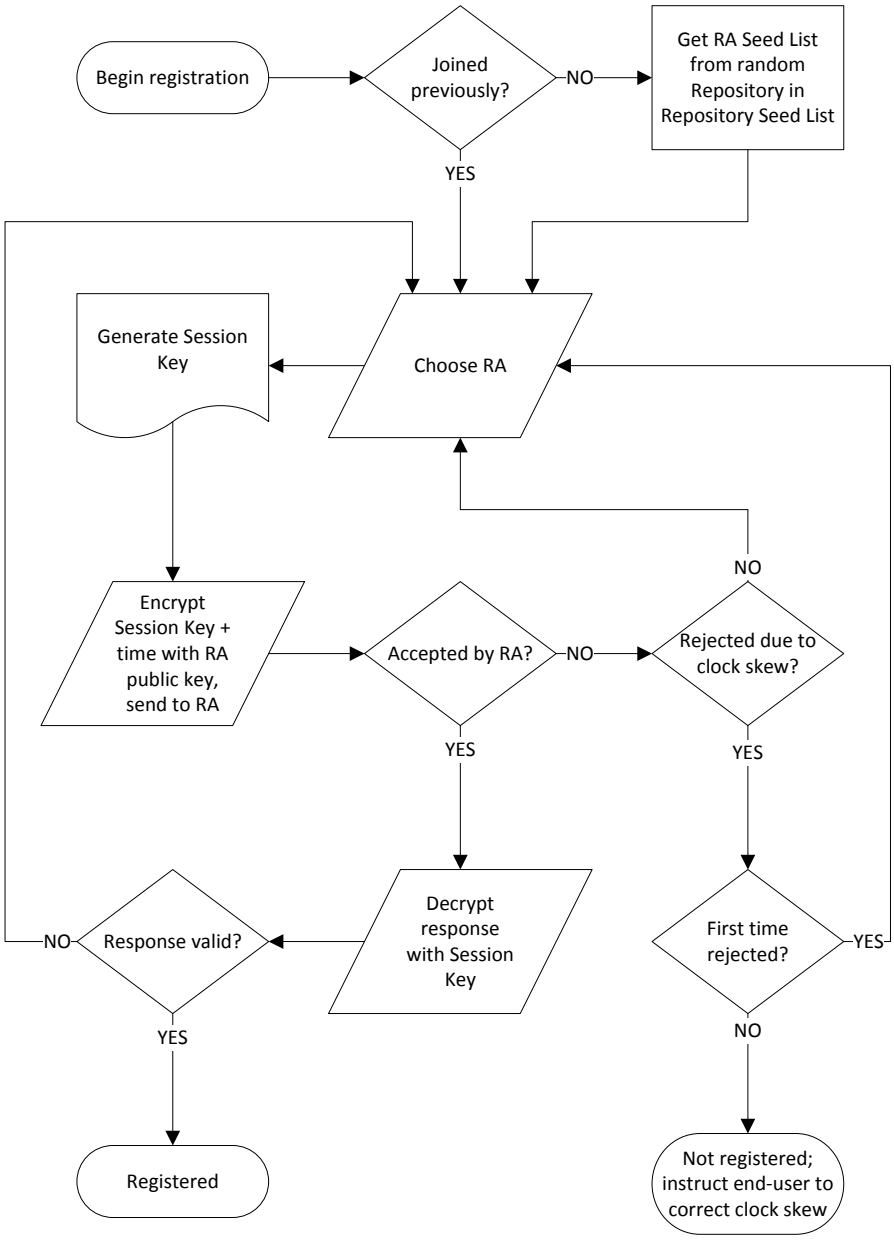

Figure 3.3 This flow diagram illustrates the steps a participant takes when attempting to join a CODON.

The `participant_leave` process provides a means for participants to gracefully leave the CODON and reduce the amount of state maintained by the most recent Regional Aggregator with which it established a session. When the participant's codonsense software service, described in Section 3.3.3, receives a graceful shutdown command from the operating system or administrator, it immediately discards any pending ISMs. The participant then notifies its

neighbors on the Sensor overlay network and its Regional Aggregator that it is shutting down by sending a `participant_leave` message. Neighboring Sensors will cease sharing or downloading any ephemeral blacklists from the departing Sensor and find a replacement neighbor on the overlay network if needed. Upon receipt of the `participant_leave` message, the Regional Aggregator immediately sets the participant's Age to the maximum value, thereby queuing the participant for deletion from the participant state database at the end of the update cycle along with any other departing or expiring participants. A clean departure from the CODON is not strictly necessary due to our use of participant aging and the implementation of peer-to-peer communication for distributing aggregated security related data. Nevertheless, a successful `participant_leave` reduces the resource burden on Regional Aggregators and reduces the time new participants may spend trying to locate peers on the overlay network because Regional Aggregators will no longer include the departed participant in seed lists. To ensure a quick shutdown, the Regional Aggregator will not send an acknowledgement, nor will the departing participant's codonsense service await one before terminating.

In the `participant_rejoin` process, the Regional Aggregator informs the participant that the maximum session age has been reached, causing the participant to invoke the `participant_join` process anew and the Regional Aggregator to purge the participant from its participant state database. This process is very similar to `participant_leave`, with the key differences being that the Regional Aggregator initiates the `participant_rejoin` process and rather than shutting down and discarding any queued ISMs, the participant's codonsense service continues monitoring and temporarily defers sending ISMs to a Regional Aggregator until the subsequent `participant_join` process is complete. Since the old state is purged from the Regional Aggregator's participant state database, participants that do not honor the

`participant_rejoin` process are penalized during subsequent update cycles as they attempt to communicate with the Regional Aggregator using their old Participant Session Key and Participant ID.

3.3.2 Repository

The purpose of the CODON Repository role is to provide useful security threat information to non-participants and to reduce the computational and network load on participants during the blacklist import and export process. As discussed earlier and previously illustrated in Figure 3.1, Regional Aggregators receive their initial seed blacklists from Repositories. Repositories also keep track of active Regional Aggregators, their Participant IDs, their public RSA keys, and Type ID subscriptions. This helps new participants locate a Regional Aggregator and helps Regional Aggregators locate peers in the Regional Aggregator overlay network and determine where to forward incoming ISMs for Type IDs to which they do not subscribe. Repositories act as the intermediary between Regional Aggregators and non-participant blacklist managers and consumers when threat information is determined by Regional Aggregators to be worth sharing during their hourly merge and update cycle.

A concise visual guide to the flow of information to and from Repositories follows in Figure 3.4. Repository content such as blacklists will likely change frequently and some non-participants may wish to directly check for updates from a CODON Repository instead of waiting for non-participant Security Information Aggregators to further refine and incorporate the new information supplied by the CODON. Repositories will publish hashes of the security data generated by the CODON, and may also transform the data into one or more formats appropriate to the systems that would make use of the information. Hashing the security data provides a means for new participants and non-participants to verify that the blacklist was

downloaded without being tampered or corrupted in transit, but we urge the use of well-established secure file transfer protocols such as HTTPS and SFTP whenever possible to provide additional assurance to the blacklist consumer that the hash and blacklist data were not both tampered (please refer to Section 2.2 for further discussion of this problem).

While a participant may serve as both a Regional Aggregator and a Repository, we believe separating these roles will allow participants who already manage internet-facing web servers to participate without the additional computational and network resources required to perform the duties of a Regional Aggregator. As published blacklist content is relatively static, i.e., it need not be stored in a database or generated and served via a dynamic script, lightweight web server software such as lighttpd [35] may be used to service blacklist requests, and common low-cost DNS-based load-balancing methods such as round-robin, geographic server distribution, and anycast [1] may be used to further reduce the burden of participating as a Repository. Repositories may also wish to redistribute additional security-related data not generated by the CODON such as attack signatures for use by an IDPS, antimalware definition updates, and open source software packages and patches, but these are not CODON functions.

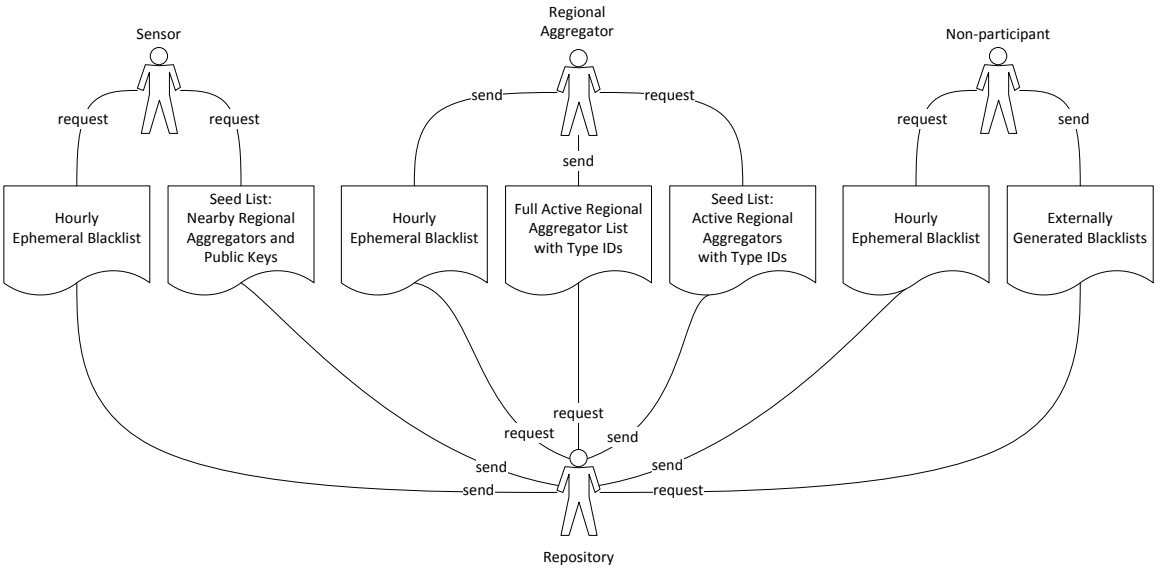

Figure 3.4 Information sharing between a Repository, CODON participants, and non-participants.

3.3.3 Sensor

Because we believe information sharing is a key component of cooperative defense, every participant of the CODON is a sensor. Sensors are responsible for sharing information with neighbors and repositories to allow for rapid collection and dissemination of intelligence about threats and ongoing attacks, underlying and overlay network health and topology changes, and recent noteworthy behaviors by fellow participants. Sensors may also subscribe to ephemeral blacklist updates and incorporate the blacklists into their system configuration for their protection. Since all participants are Sensors, each participant can potentially contribute to the health of the CODON and the internet at large when malicious activity is detected.

The duties of a Sensor can be subdivided into two software services: codonsense and codonblock, illustrated below in Figure 3.5. The codonsense service is responsible for Sensor-to-Regional Aggregator session management and for gathering recent security incident data from any available security data sources. These recent data are then reported to the Sensor's corresponding Regional Aggregator by codonsense. Historical data with a timestamp older than

two update cycles (ten minutes) is to be completely disregarded since Regional Aggregators only consider newer incidents when updating their ephemeral blacklists. Data transformation plugins will be necessary to ensure that codonsense can parse and convert the data from various data sources for use in the CODON. The transformation plugins should be distributed with the Sensor software package for ease of use.

The codonblock service is an optional service responsible for applying ephemeral blacklists to the Sensor's security systems. We anticipate that home users with the codonblock service enabled may wish to apply all new ephemeral blacklist entries to their systems immediately. In contrast, participants with larger and more complex networks may wish only to apply blacklist updates at certain times throughout the day due to organizational change management policies. Participants may also wish to use different thresholds for determining whether to block access based on an ephemeral blacklist entry. For example, a participant mainly interested in safe web browsing may wish for all entries in an ephemeral blacklist to apply immediately, whereas a participant whose primary concern is malicious systems attempting to compromise an e-commerce web server may wish to be more conservative to avoid losing potential customers. The latter participant may choose only to block access to the web server if an entry has an expiration of at least 24 hours, implying that the host or network in the blacklist entry was reported as malicious by multiple Sensors over a period of time.

Any security-related data source may be used as a source of security incident data to be monitored and shared by the CODON Sensor software with the use of an appropriate data source parser. While the Sensor software should be able to gather security incident data from hardware firewall logs and other common devices via standardized and common methods like SNMP (Simple Network Management Protocol) and a syslog listener, locally generated and

stored logs in the form of plaintext files or operating system specific log formats may also be leveraged via a custom data source parser. Custom data source parsers can be useful for extracting security-related information from proprietary software that may not use common or standard logging formats. Custom data source parsers should be vetted for correctness by technically skilled participants in the CODON community and distributed with the CODON software for the benefit of future participants when possible.

The use of an IDPS, enterprise-grade firewall, or other advanced security software can undoubtedly provide useful security incident information for the CODON, but it is not necessary for participation as a Sensor. Home networks with ISP-provided multifunction modems and router/firewall devices as well as host-based firewalls will likely provide the majority of data reported to Regional Aggregators. Most modern home computer operating systems include host-based firewalls, and these firewalls are usually enabled by default. Host-based firewalls include the ubiquitous netfilter [6] included with mainstream Linux distributions, the eponymous Windows Firewall included with Microsoft Windows client and server operating systems, and various third-party firewall software systems that can be purchased and/or downloaded over the Internet.

Most Sensors will only be able to report passively observed security incident information on a small scale, but some Sensors have unique resources enabling them to passively detect larger scale events or proactively detect security issues with other systems. We refer to the latter category of Sensors as “Active Detectors.” Large scale passive detection capabilities are usually limited to participants with unique resources or unique topographical placement on the internet, such as internet service providers, transit providers, and CDNs, but also include research projects like the UCSD Network Telescope [14] and University of Oregon’s

Route Views [70] project. Active Detectors may offer network perimeter vulnerability scanning services [69] and/or auditing and validation of security best practices. Since Active Detectors engage in activity that may be considered malicious or suspicious in the course of their operations, they must register with a Defensive Service Broker (described in Section 3.3.4) before engaging in active detection functions like port scanning to ensure they are not blacklisted. To ensure that security incident reports from a Sensor with large scale passive detection capabilities is given a proportionate level of credibility, the Sensor's administrator may choose a unique Type ID and encourage other participants to give the corresponding ephemeral blacklist a greater weight as described in Section 3.4.

Reporting security incidents such as local virus infections or successful intrusions is encouraged as the information can be very valuable to the CODON, helping research organizations and the CODON itself identify emerging threats and attacks. However, sharing information about one's own system or network can introduce a risk of attack by traitorous participants and outsiders, so by default Sensors only share information about external-to-internal network based security incidents. Furthermore, when a Sensor reports a security incident to a Regional Aggregator, the default behavior is for the Sensor to omit the specific destination/victim IP and any private IP address ranges to avoid divulging information about the Sensor's local network, leaving the Regional Aggregator to assume that the Sensor's WAN IP address was the destination. Our use of session encryption also helps ensure that only Regional Aggregators, not participants serving solely as Sensors, receive the sensitive details of an attack. We believe this behavior strikes a reasonable balance for most participants between providing credible threat information while preserving participant privacy, but Sensors will likely choose only to omit private IP range information. For example, a Sensor may use an intrusion detection

system to monitor a variety of hosts with one or more WAN IPs on its network, and providing the attack destination IPs will present a clearer picture of the scope of a distributed attack by reporting the various WAN IPs of the attacked hosts.

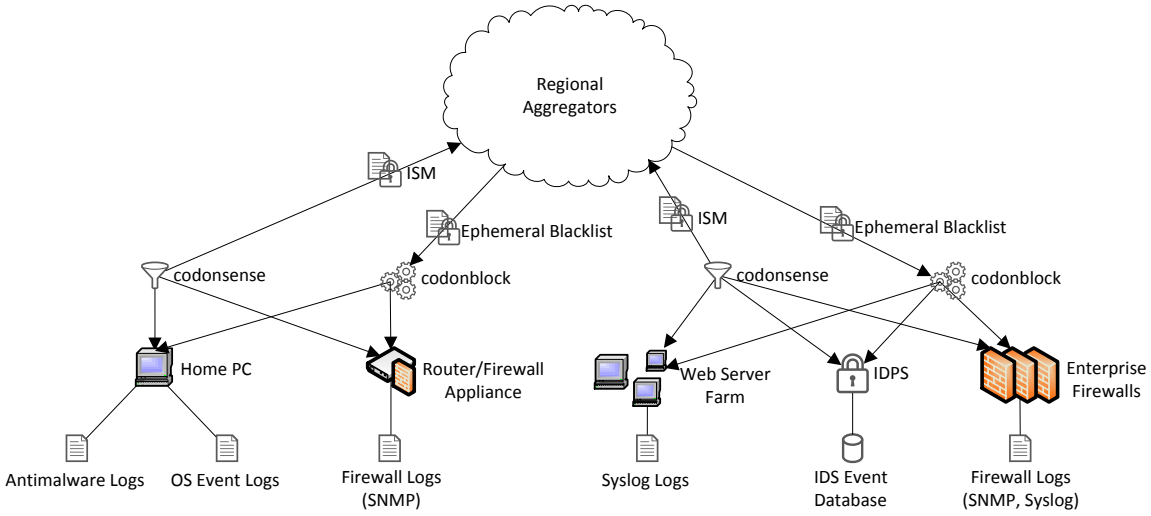

Figure 3.5 A home user (left) and a web hosting provider (right) particiate as Sensors with the optional codonblock service enabled. Any available source of security data may be monitored by the codonsense service.

3.3.4 Defensive Service Broker

It is often desirable to assess one's own network's security by performing an external test from another network. This offers the administrator an outsider's perspective of the network and may reveal security issues that are difficult to discern internally. Defensive Service Brokers, or simply "Brokers," serve as matchmakers for participants seeking proactive detection services and other parties offering such services including periodic vulnerability scanning and one-time or recurring intensive security assessments such as penetration tests. Such network security activities are classified as malicious when unsolicited, so it is necessary to provide a mechanism by which security service providers can continue contributing to fellow participants' security using their unique resources without being blacklisted.

While a participant could simply ask a neighboring CODON participant to externally assess the security of its network, we expect the majority of participants to lack proactive detection abilities. Legal contracts are often used to set parameters and boundaries for penetration testing, as security assessments performed by third parties require a level of trust to be established between the tester and the requesting party. Security service providers may also wish to perform their activities from non-participant networks and systems to better simulate real-world security threats. Brokers notify repositories in advance of the agreed-upon but potentially suspicious activity based on specific source and destination IP address ranges and the type of behavior agreed upon by the service provider and requesting party.

Service requesters and service providers such as Active Detectors register requests and proactive security offers with a Broker via ISMs, and when a Broker finds a match that would satisfy both parties' criteria, the Broker notifies both parties, who acknowledge and agree to the match. Both parties must agree before the matchmaking opportunity expires. The Broker then notifies Regional Aggregators of the upcoming activity along with a time window indicating when the activity is expected to take place. Based on this information, Regional Aggregators will ignore any reports of suspicious activity between the participants during specific time windows. The applicable time window will be updated if the service request is completed early. If a match is successful, the service provider may earn reputation with the participant or multiple participants. If a match is unsuccessful because the service provider exceeded the time window or failed to complete the requested service, that party's reputation suffers as participants send ISMs with the appropriate Type ID to their Regional Aggregators noting the activity (or inactivity, as the case may be).

3.4 Information Sharing Messages and Scoring

The sharing of security incident information from Sensors to Regional Aggregators occurs via Information Sharing Messages (ISMs). ISMs are simply encrypted messages containing the sender's Participant ID and incident information in a common format appropriate to the message's subject matter. For example, an ISM regarding an excessive number of unsuccessful attempts to log into a website in a short period of time might include the CODON participant's Participant ID, the Type ID of the blacklist corresponding to the type of attack, the source IP of the attack, the destination port of the attack, and the time the attack was logged or blocked by the participant's system. The Type ID encapsulated in the ISM is essential for a Regional Aggregator to determine whether it should consider the ISM when updating an ephemeral blacklist that it manages, or if it should instead pass the ISM to other interested Regional Aggregators in its overlay network. As discussed above in Section 3.3.1.3, encrypting ISMs is done primarily for non-repudiation and to reduce the threat of eavesdroppers deceiving other participants via counterfeit messages. We illustrate the process of sending an ISM in Figure 3.6.

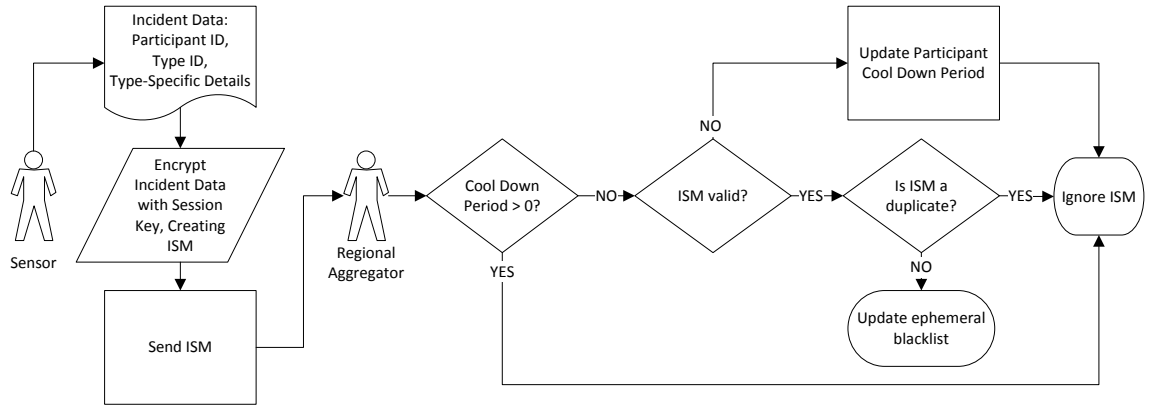

Figure 3.6 A Sensor sends an Information Sharing Message (ISM) to its Regional Aggregator.

3.4.1 Evaluating the Costs of Different Threats

It is difficult to quantify the costs and benefits of network defense and health activity, but we have chosen three useful evaluation criteria when considering how one might participate

in a CODON, as well as how one might respond to actions that occur during the course of CODON participation. To respect the heterogeneity and differing priorities of a CODON, these criteria can be weighted by individual participants based on what they deem to be important. The criteria we have selected are monetary cost change (Δc), network health / defensive posture change (Δh), and reputation change (Δr). While the goal of most participants will be maximizing Δh while minimizing Δc with a side effect of increasing Δr, participants with more resources to contribute to the CODON will likely seek to maximize Δr through spending money (Δc) to increase Δh and as a side effect garner more public attention, potential customers, research funding, etc.

While all CODON participants are required to accept and propagate new security incident information with fellow participants in the peer-to-peer overlay network, each participant may choose to "subscribe" to particular incident Type IDs via the codonblock service. This allows the participant to apply weights to each incident of a particular type and use this weight information to maintain a local blacklist with expiration durations tailored to the participant's desires. This may be useful for large web server farms that only wish to block compromised home users for a brief time or serve a custom webpage directing the same users to a non-participant Remediator. On the other hand, many home users and corporate network administrators may wish to begin blocking websites that distribute malware as soon as they are reported instead of waiting for a large number of corroborating reports from other CODON participants.

3.4.2 Assigning Scores to Consequences

Actions by CODON participants have consequences, and in our model these consequences take the form of a three-tuple $(\Delta c, \Delta h, \Delta r)$ where each element of the tuple

corresponds to the aforementioned evaluation criteria of monetary cost change, network health/defensive posture change, and reputation change. Default consequence scores are assigned to behavior classes as described in Section 3.4.4, and are to be maintained and distributed with the CODON software. Each element of the consequence tuple may range from -100 to 100. Participants convert consequence to single values for each behavior class based on how strongly the participant feels about the behavior. The resulting single value score can be used to determine trends in individual participants' behavior, how valuable a certain participant's participation may be, or whether to adjust ephemeral blacklist expirations to better meet the participant's desires. Public relations departments in large organizations may wish to use this scoring system to advertise their benevolence in year-end charitable activity reports, and small organizations and home users may likewise wish to advertise their benevolence for competitive or reputational benefit on social media networks. Similarly, Repositories and non-participant researchers may perform statistical and historical analysis of the volume and quality of CODON contributions compared to other research efforts to evaluate the effectiveness of the CODON and refine their own blacklist criteria.

3.4.3 Actor Classification

CODON participants and non-participants can be classified based on historical behavior. We will refer to both participants and non-participants in general as "actors" when discussing classification. A CODON by its nature is best served when all participants consistently classify a particular actor as having the same actor class, and therefore interact with that actor accordingly. Nevertheless, an individual participant may choose to classify a particular actor differently due to recent events not yet communicated throughout the CODON or due to circumstances only relevant to that individual participant. It is also possible that actors who

consistently engage in friendly behavior may be treated with distrust by other actors due to ideological or geopolitical differences. Table 3.5 lists actor classes in this model and their defining characteristics and behaviors.

Table 3.5 CODON actor classifications

Actor Class	Participant?	Defining Characteristics and Historical Behaviors
Good Neighbor	Yes	Has a history of positive actions/contributions in the CODON.
Protectorate	Yes	Has only the Sensor role and a short (or no) history of actions/contributions in the CODON.
Pariah	Yes	Participant with a history of predominantly negative actions. A "three strikes" rule or negative value reputation threshold may be employed to differentiate pariahs from participants that are experiencing a temporary crisis.
Offender	No	Has a history of predominantly negative actions as evidenced by inclusion in one or more blacklists.

3.4.4 Behavior Classes and Evaluation Criteria for Scoring

Both friendly and unfriendly actions are worth considering when evaluating how an actor should be classified and what an appropriate response should be for different CODON participants. Actions of interest to participants may be divided into several behavior classes. Table 3.6 lists these behavior classes along with corresponding example behaviors and some possible default consequence scores as discussed in Section 3.4.2.

Table 3.6 Behavior classes and scoring of CODON participant actions

Behavior Class	Example(s)	Δc	Δh	Δr
Detection	Vulnerability scanning; firewall and IDPS log aggregation and sharing; network monitoring for malicious traffic flows; monitoring sudden internet topology changes	15	20	10
Unsolicited Reconnaissance	Performing active detection functions without first agreeing via a Broker; failing to perform Brokered services	-10	-25	-25
Information Propagation	Successfully and consistently sharing ISMs and ephemeral blacklists via the peer-to-peer overlay network; participating as a Repository	5	5	5
Withholding Information	Not sharing ephemeral blacklists via the overlay network, e.g. being blocked by egress firewall filtering, CODON software malfunctioning	-5	-10	-5
Incident Self-Reporting	Sharing detailed IODEF [16] incident reports about malware outbreaks in one's own network via ISMs; sharing incident reports about attempted attacks	10	30	25
Remediation Provision	Hosting software patch repositories (Linux distribution mirrors, etc.); Common Remediation Enumeration (CRE [42]) repository publishing	30	30	25
Remediation Implementation	Software patch or anti-malware definition development; CRE creation; updating blacklists; malicious network classification; throttling attack traffic	30	30	30
Malicious Activity	Participating in a DDoS; sending spam or phishing email; appearing on a blacklist	-40	-40	-40

3.4.5 Scoring and Weighting Example

Consider a home user who is scanned for vulnerabilities by another participant having not first agreed to the scan via a Defensive Service Broker, and a small web hosting service in the same circumstance. Consequences are weighted and scored using the following formula:

$$S = \frac{w_c * \Delta c + w_h * \Delta h + w_r * \Delta r}{\Delta c + \Delta h + \Delta r}$$

Suppose the home user finds the default consequence scores to be adequate and weighs all consequence elements equally using CODON's default weight of $(w_c, w_h, w_r) = (1, 1, 1)$ but the small web hosting service thinks monetary cost is twice as important as the other evaluation criteria and thus adjusts all consequences using a weight of $(w_c, w_h, w_r) = (2, 1, 1)$. Using the scoring formula above and the default unsolicited reconnaissance consequence of $(-10, -25, -25)$, the home user would score this behavior as 1, but the web host would score it as 1.17. If the unsolicited scanning activity were to continue, the web host

would locally blacklist the offender via the codonblock service sooner and for a longer duration than the home user and other participants might.

3.5 Potential for CODON Abuse

A malicious actor may foreseeably use a botnet to install a tampered version of the CODON software to falsely report a victim as being malicious and undermine the CODON's trustworthiness. The use of information expiration, or the "ephemeral" in ephemeral blacklists, helps ensure that if an attempt to poison the quality of information generated by the CODON is successful, the attack's effectiveness will be limited. Smaller scale attempts to falsely report a victim are unlikely to succeed due to the requirement that multiple Sensors must report malicious activity in a short period of time for the threat to be considered credible. Weighting particular threats and blacklists by Type ID can further mitigate the botnet threat, but a whitelist may be more appropriate to override blacklist settings. For example, it may be beneficial to whitelist well-known e-commerce websites and non-participant Security Information Aggregators and Remediators. Whitelisting may leave participants vulnerable if one of the whitelisted hosts are compromised, and whitelists can be very burdensome to manage as we discussed in Section 2.10, particularly when many organizations use third-party CDNs or own multiple discontiguous blocks of IP addresses across multiple ASes that could change frequently.

While we have proposed a numerical scoring system for actions based on their perceived friendliness, we admit that no scoring system will perfectly reflect the complex and dynamic nature of cross-organizational relationships, and most home users may find the weighting mechanism confusing. When a fellow CODON participant behaves in such a way that merits a classification change, we believe particularly with larger organizations that CODON-generated intelligence should be used as part of the human decision making process, not as a

substitute. Periods of attacks from another participant's network may not always justify blacklisting that participant from the CODON and reclassifying the participant as an offender.

3.6 Barriers to Adoption and Some Watershed Moments

With a distributed system on the scale of the internet, widespread infrastructure changes seldom occur quickly; one needs only consider IPv6, the long-awaited successor to IPv4. Among other benefits, IPv6 uses 128 bit IP addresses compared to the 32 bit addresses used in IPv4. Excluding reserved blocks of private addresses, 2^{128} addresses can be used in IPv6, whereas only 2^{32} can be represented in IPv4. Despite the long-predicted and impending depletion of available IPv4 addresses, administrators have gone to great lengths to avoid costly upgrades and learning a new and complex system. This is despite the existence of early IPv6 implementations in the mid-to-late-1990s [33] and its inclusion in all modern operating systems, although adoption has increased significantly in the past two to three years through the efforts of a consortium of major hardware, software, and service vendors [34]. Port address translation (PAT), often mistaken for and commonly referred to network address translation (NAT), allows multiple privately-addressed IPs to communicate through a single public IP address. This serves as a cheap and ubiquitous stopgap solution to the problem of dwindling publicly addressable IP addresses and creates added layers of configuration complexity and security as privately-addressed hosts can easily initiate communication to publicly-addressed hosts, but the reverse is difficult. For infrastructure providers, cost and profit tend to be primary drivers for major upgrades, but for home users the usefulness or "wow factor" of a service tend to drive upgrades. Considering the rise and trends of mobile computing over the past several years, particularly the shift of smartphones from business users to casual/novice internet users, we believe tailoring the CODON software toward less technical end-users will have the greatest

probability success. As home users experience the benefit of a safer internet and share this experience with their real and virtual communities, larger organizations will be attracted to this momentum and participate in the more resource intensive CODON roles of Regional Aggregator and Repository. With greater participation from larger organizations, we expect the CODON architecture to undergo refinement and optimization, evolving to better serve the internet community.

There are also cultural and political barriers that may affect the growth and adoption of CODONs. Culturally speaking, many technical users may resist sharing what they consider private metadata about their network, especially in light of recent revelations about state-sponsored metadata gathering and domestic spying. The open source spirit of the CODON framework should help allay those fears to some degree, as technical users can review and verify the CODON software's behavior. Furthermore, the CODON's emphasis is on detection of malicious and suspicious security-related behavior from other internet users, not one's own web browsing or other communication habits. Politically speaking, some nations and organizations ban the use of strong encryption, or prohibit its use between internal networks and external networks to enforce censorship policies or defend against sensitive data exfiltration. Such nations may wish to modify the CODON software and operate their own CODONs that comply with political requirements. While we consider the prohibition of encryption and the existence of censorship deeply regrettable, the existence of a small number of politically-driven CODONs may still contribute to improving security for the internet community as a whole.

Regional Aggregators and Repositories are responsible for managing and sharing numerous blacklists based on varying criteria via Type IDs, and the resource commitment for these participants may be significant, especially as a CODON grows. We do not believe this to be

a significant barrier in light of the proliferation of ever-scaling high performance and commodity "cloud computing" services provided by Amazon, Microsoft, and others. We do not believe resource utilization by home users will pose a barrier to CODON participation, as projects such as Stanford University's Folding@home [52], virtual currency systems based on computing meaningless hashes like Bitcoin [8], and BitTorrent-based file sharing are widely used even among less technically adept members of the internet community despite their significant resource requirements.

We see a great opportunity to spur CODON participation through the cooperation of internet service providers and consumer-grade network security appliances. Several consumer firewall hardware manufacturers have in recent years built upon the efforts of open source custom firmware developers to inexpensively include more features in their equipment and appeal to enthusiasts, and ISPs commonly include a software CD and free or subsidized subscriptions to antimalware software for new customers. By including CODON Sensor software in consumer-grade security devices and ISP customer software bundles, these manufacturers and ISPs could significantly expand the number of Sensors participating in the CODON. Using the incentive and consequence scoring paradigm we detailed throughout Section 3.4, manufacturers may see an improvement in reputation and ISPs may see an improvement in both reputation and network health/defensive posture as their customers' systems are less frequently compromised. Home computer manufacturers could further promulgate the CODON Sensor software through the common practice of pre-installing the software at the factory.

With its heavy bent toward open source, the CODON may initially take greater hold among the Linux community than among proprietary operating system users. As such, designing the CODON Sensor software as a signal-aware daemon, to use kernel modules for performance

acceleration, or with methods to directly interface with specialized network hardware or the netfilter firewall [6] may build momentum among Linux home users, developers, and enterprise system administrators by encouraging various Linux distribution maintainers to include the software as a useful core or optional component.

4 Conclusion

We have described a framework for developing and deploying a global crowdsourced defensive network to make automated network-based threat detection, information sharing, and defense accessible to the masses. As malicious actors continue to find innovative ways to wreak havoc against unsuspecting internet users on an unprecedented scale, the need for a more panoramic view of network-based threats is clear. It is also clear to us that cooperative defense strategies hold the most hope for effective defense against numerous cooperative aggressors. While large commercial organizations such as major software vendors stand to gain less defensive benefit from participation in a CODON than home end-users, their reputations are likely to improve as their customers and potential customers benefit from timely threat information sharing and a history of being a "good neighbor" in the wild and dangerous internet. We believe that even a low CODON participation rate can have a noticeable positive effect on the internet by providing quick and actionable intelligence to those system administrators with the unique resources and specialized tools necessary for mitigating distributed attacks. Even the formation of many small CODONs based on differing geopolitical, ideological, and commercial motivators would provide a benefit to the larger internet community as different "neighborhoods" of the internet become safer.

5 Future Work

We would like to see a CODON come to fruition through the implementation of the various CODON roles in software and collaborative testing between the academic research, private research, and online enthusiast communities. And as we already discussed in Section 3.6, building and optimizing the CODON software for inclusion and use in home firewall/router firmware would surely go a great way toward making the CODON dream a reality. The CODON software could be further refined to understand and comply with mandatory access controls and security policies on systems using SELinux [44] to provide further assurance to system administrators that sensitive internal data will not inadvertently be shared with the larger CODON community. We provide a list of milestones that should take place to bring CODON from concept to successful widespread usage in Table 5.1.

Table 5.1 Major milestones in the process of taking CODON from concept to widespread adoption.

Milestone	Description
M1	CODON roles implemented in software using cross-platform languages and libraries with minimal set of data transformation plugins.
M2	CODON successfully tested using multiple internet-facing hosts within the same organization.
M3	CODON successfully tested using multiple internet-facing systems across multiple academic/research organizations, adding data transformation plugins to interface with participants' security systems.
M4	CODON registered as a new project at a publicly accessible open-source project repository, made available for common operating systems.
M5	Research and enthusiast communities adopt CODON, contributing and testing useful data transformation plugins.
M6	CODON earns reputation as a useful and mature security system.
M7	CODON software included by default in Linux distributions and incorporated into ISP-provided router firmware.

While we believe a publicly accessible CODON will benefit the largest segment of the internet, we also acknowledge that governments, regulatory agencies, sensitive research labs, and other entities with special requirements may wish to employ a federated approach whereby

only select participants may participate, let alone fill certain roles. Such entities may also wish to integrate existing authentication and authorization systems, or even collect administrative fees from participants to pay for dedicated administrators and security data analysts to clean more insightful information than strictly blacklists from their more selective CODONs. The CODON framework could be extended to incorporate federated participation and authentication.

The CODON framework as we have described it does not explicitly support IPv6. This was done for the sake of simplicity and because as we alluded to in Section 3.6, IPv6 still faces an uphill battle for full adoption and remains unfamiliar and confusing to many administrators. CODON could be extended with some effort to incorporate IPv6 concepts such as site-local and link-local networks and the various IPv6-to-IPv4 bridging and tunneling protocols.

References

[1] Abley, J., Lindqvist, K. (Dec. 2006). Operation of Anycast Services. IETF, RFC 4786, BCP 126. Available: http://www.ietf.org/rfc/rfc4786.txt. Accessed: Mar. 2014.

[2] Akamai Technologies. (2014). About | Akamai. Available: http://www.akamai.com/html /about/index.html. Accessed: Feb. 2014.

[3] Anderson, R. J. (2008). Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed. Chichester, U.K. John Wiley and Sons, Ltd.

[4] Anti-Abuse Project, The. (2012). Multi-RBL Check. Available: http://www.anti-abuse.org /multi-rbl-check/. Accessed: Feb. 2014.

[5] Ashton, K. (Jun. 2009). That "Internet of Things" Thing. RFID Journal. Available: http://rfidjournal.com/articles/view?4986. Accessed: Jan. 2014.

[6] Ayuso, P. N. (2010). netfilter/iptables project homepage - The netfilter.org project. Available: http://www.netfilter.org/. Accessed: Mar. 2014.

[7] Beckett, W., Thusitha, J. (Nov. 2011). Botnet-Originated DDOS Attacks and their Mitigation – A New Spiral in the Arms Race on the Internet. Global Science and Technology Forum. Available: http://www.research.att.com/export/sites/att_labs/techdocs/TD_100483.doc. Accessed: Feb. 2014.

[8] Bitcoin Project. (2014). Bitcoin - Open source P2P money. Available: https://bitcoin.org/. Accessed: Mar. 2014.

[9] Blizzard Entertainment, Inc. (2014). How to Toggle Peer to Peer Protocol - Battle.net Support. Available: https://us.battle.net/support/en/article/how-to-toggle-peer-to-peer-protocol. Accessed: Feb. 2014.

[10] Carts, D. A. (Nov. 2001). A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols. SANS Institute Reading Room. Available: http://www.sans.org/reading-room/whitepapers/vpns/review-diffie-hellman-algorithm-secure-internet-protocols-751. Accessed: Mar. 2014.

[11] Cisco *et al.* (2014). Snort :: About Snort. Available: http://www.snort.org/snort. Accessed: Feb. 2014.

[12] CloudFlare, Inc. (2014). An Overview of CloudFlare. Available: https://www.cloudflare.com /overview. Accessed: Feb. 2014.

[13] Cohen, B. (Oct. 2012). The BitTorrent Protocol Specification. Available: http://bittorrent.org/beps/bep_0003.html. Accessed: Feb. 2014.

[14] Cooperative Association for Internet Data Analysis, The. The UCSD Network Telescope. Available: http://www.caida.org/projects/network_telescope/. Accessed: Apr. 2014.

[15] Cunningham, W. Wiki Wiki Web. Available: http://c2.com/cgi/wiki?WikiWikiWeb. Accessed: Dec. 2013.

[16] Danyliw, R., Meijer, J., and Demchenko, Y. (Dec. 2007). The Incident Object Description Exchange Format. IETF, RFC 5070. Available: http://www.ietf.org/rfc/rfc5070.txt. Accessed: Apr. 2014.

[17] Dice Holdings, Inc. (2013). SourceForge - Download, Develop and Publish Free Open Source Software. Available: http://sourceforge.net/. Accessed: Dec. 2013.

[18] Dierks, T., Rescorla, E. (Aug. 2008). The Transport Layer Security (TLS) Protocol Version 1.2. IETF, RFC 5246. Available: http://tools.ietf.org/rfc/rfc5246.txt. Accessed: Apr. 2014

[19] EasyList Project. EasyList and EasyPrivacy Policy. Available: https://easylist.adblockplus.org /en/policy. Accessed: Jan. 2014.

[20] Eiler, D. (Apr. 2012). Key Exchange Methods, CPE 701 Research Case Study. Available: http://www.cse.unr.edu/~derek/cpe701/keyex.pdf. Accessed: Mar. 2014.

[21] Eiler, D. (Apr. 2012). Key Exchange Methods: Diffie-Hellman and RSA. Available: http://www.cse.unr.edu/~derek/cpe701/keyex.pptx. Accessed: Mar. 2014.

[22] Elisa, B., Ferrari, E., Squicciarini, A.C. (2004). Trust-X: a peer-to-peer framework for trust establishment. IEEE Transactions on Knowledge and Data Engineering 16.7, pp. 827-842.

[23] Eyeo GmbH. About AdBlock Plus. Available: https://adblockplus.org/en/about. Accessed: Jan. 2014.

[24] Fink, E., Sharifi, M., Carbonell, J.G. (2011). Application of Machine Learning and Crowdsourcing to Detection of Cybersecurity Threats. Proceedings of the US Department of Homeland Security Science Conference - Fifth Annual University Network Summit.

[25] Frank, B. *et al.* (Jul. 2013). Pushing CDN-ISP Collaboration to the Limit. ACM SIGCOMM Computer Communication Review, Vol. 43, Issue 3, pp. 35-44.

[26] Gergely, A. “Irish student’s Jarre wiki hoax dupes journalists.” Thomson Reuters, 7 May, 2009. Available: http://www.reuters.com/article/2009/05/07/us-wikipedia-hoax-idUSTRE5461ZJ20090507. Accessed: Mar. 2014.

[27] Gillett, M. (Jul. 2012). What Does Skype’s Architecture Do? Available: http://blogs.skype.com/2012/07/26/what-does-skypes-architecture-do/. Accessed: Dec. 2013.

[28] Gillett, M. (Oct. 2013). Technological Changes to Improve the Skype Experience. Available: http://blogs.skype.com/2013/10/04/skype-architecture-update/. Accessed: Dec. 2013.

[29] GitHub, Inc. (2013). GitHub - Build software better, together. Available: https://github.com/. Accessed: Dec. 2013.

[30] Google. (8 May, 2013). Safe Browsing API - Google Developers. Available: https://developers.google.com/safe-browsing/. Accessed: Mar. 2014.

[31] Grothoff, C. *et al.* GNU's Framework for Secure Peer-to-Peer Networking. Technische Universität München. Available: https://gnunet.org/. Accessed: Mar. 2014.

[32] Guttman, B., Roback, E. A. (Oct. 1995). *An Introduction to Computer Security: The NIST Handbook (Special Publication 800-12)*. National Institute of Standards and Technology. Available: http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf. Accessed: Apr. 2014.

[33] International Business Machines. IPv6 at IBM. Available: http://www-01.ibm.com/software/info/ipv6/index.jsp. Accessed: Mar. 2014.

[34] Internet Society, The. World IPv6 Launch. Available: http://www.worldipv6launch.org/. Accessed: Mar. 2014.

[35] Kneschke, J. (2014). Home - Lighttpd - fly light. Available: http://www.lighttpd.net/. Accessed: Feb. 2014.

[36] Leach, P., Mealling, M., Salz, R. (Jul. 2005). A Universally Unique Identifier (UUID) URN Namespace. IETF, RFC 4122. Available: http://www.ietf.org/rfc/rfc4122.txt. Accessed: Feb. 2014.

[37] Locasto, M.E., Parekh, J.J., Keromytis, A.D., Stolfo, S.J. (2005). Towards Collaborative Security and P2P Intrusion Detection. Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, pp. 333-339.

[38] Lyon, G. Nmap - Free Security Scanner for Network Exploration & Security Audits. Available: http://nmap.org/. Accessed: Mar. 2014.

[39] Mirkovic, J., Robinson, M., Reiher, P. (2003). Alliance formation for DDoS defense. Proceedings of the 2003 workshop on new security paradigms, pp. 11-18.

[40] MITRE Corporation, The. (2014). CVE - Common Vulnerabilities and Exposures. Available: http://cve.mitre.org/. Accessed: Mar. 2014.

[41] MITRE Corporation, The. (2014). OVAL - Open Vulnerability and Assessment Language. Available: http://oval.mitre.org/. Accessed: Mar. 2014.

[42] National Institute of Standards and Technology. (2014). CRE - Common Remediation Enumeration. Available: http://scap.nist.gov/specifications/cre/. Accessed: Mar. 2014.

[43] National Institute of Standards and Technology. (2014). The Security Content Automation Protocol (SCAP). Available: http://scap.nist.gov/. Accessed: Mar. 2014.

[44] National Security Agency, Central Security Service (Jan. 2009). SELinux Frequently Asked Questions (FAQ) - NSA/CSS. Available: http://www.nsa.gov/research/selinux/faqs.shtml. Accessed: Mar. 2014.

[45] Nazario, J. (Apr. 2009). PhoneyC: A virtual client honeypot. Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats. USENIX Association. Available: https://www.usenix.org/legacy/event/leet09/tech/full_papers/nazario/nazario.pdf. Accessed: Mar. 2014.

[46] NewEraCracker. (Apr. 2013). LOIC/Readme. Available: https://github.com/NewEraCracker/LOIC/blob/master/README. Accessed: Feb. 2014.

[47] Oikonomou, G., Reiher, P., Robinson, M., Mirkovic, J. (Dec. 2006). A Framework for Collaborative DDoS Defense. Proceedings of the 2006 Annual Computer Security Applications Conference (ACSAS 22).

[48] Open Internet Security Foundation, The. Suricata | Open Source IDS / IPS / NSM engine. Available: http://suricata-ids.org/. Accessed: Mar. 2014.

[49] OpenBL.org Project. OpenBL.org - Abuse Reporting and Blacklisting. Available: https://www.openbl.org/faq.html. Accessed: Jan. 2014.

[50] OpenSSL Project, The. (2014). OpenSSL: The Open Source toolkit for SSL/TLS. Available: https://www.openssl.org/. Accessed: Feb. 2014.

[51] OverClocked ReMix, LLC. (Jan. 2013). Torrents - OCRWiki - OverClocked ReMix. Available: http://ocremix.org/wiki/index.php?title=Torrents&oldid=5009. Accessed: Nov. 2013.

[52] Pande, V. (2013). Folding@home. Available: https://folding.stanford.edu/. Accessed: Mar. 2014.

[53] Peng, T., Leckie, C., Ramamohanarao, K. (Mar. 2007). Survey of network-based defense mechanisms countering the DoS and DDoS problems. ACM Computing Surveys, Vol. 39 (1).

[54] Proofpoint, Inc. (2013). SORBS (Spam and Open-Relay Blocking System). Available: http://www.sorbs.net/. Accessed: Dec. 2013.

[55] Provos, N. (2004). A Virtual Honeypot Framework. USENIX Security Symposium, Vol. 173. USENIX Association. Available: http://static.usenix.org/event/sec04/tech/full_papers/provos/provos_html/. Accessed: Mar. 2014.

[56] Rapid7. Vulnerability Management & Risk Management Software | Rapid7. Available: http://www.rapid7.com/products/nexpose/. Accessed: Mar. 2014.

[57] Research and Education Networking Information Sharing and Analysis Center. (Sep. 2009). REN-ISAC Information Sharing Policy. Available: http://www.ren-isac.net/docs/information_sharing_policy.html. Accessed: Mar. 2014.

[58] Schwartz, M.J. (4 Oct. 2013). Operation Payback: Feds Charge 13 On Anonymous Attacks. InformationWeek. Available: http://www.informationweek.com/attacks/operation-payback-feds-charge-13-on-anonymous-attacks/d/d-id/1111819. Accessed: Feb. 2014.

[59] SC-integrity. (2007). About the SC ISAC. Available: https://secure.sc-investigate.net/SC-ISAC/ISACAbout.aspx. Accessed: Mar. 2014.

[60] Sharifi, M., Fink, E., Carbonell, J.G. (Oct. 2011). SmartNotes: Application of crowdsourcing to the detection of web threats. 2011 IEEE International Conference on Systems, Man and Cybernetics (SMC), pp.1346, 1350.

[61] Shue, C.A., Kalafut, A.J., Gupta, M. (2011). Abnormally Malicious Autonomous Systems and their Internet Connectivity. IEEE/ACM Transactions on Networking 20.1, pp. 220-230.

[62] Skype, Microsoft. (2014). What is Skype? Learn all about Skype's free and low-cost features. Available: http://www.skype.com/en/what-is-skype/. Accessed: Feb. 2014.

[63] Software Engineering Institute, Carnegie Mellon University. About Us | The CERT Division. Available: https://www.cert.org/about/. Accessed: Feb. 2014.

[64] Software in the Public Interest, Inc. (Dec. 2013). Downloading Debian CD images with BitTorrent. Available: http://www.debian.org/CD/torrent-cd/. Accessed: Jan. 2014.

[65] Spafford, E. H. (Nov. 1989). The Internet Worm Program: An Analysis. ACM SIGCOMM Computer Communications Review, Vol. 19, Issue 1, pp. 17-57.

[66] Spamhaus Project Ltd., The. (1998-2014). The Spamhaus Project - Understanding DNSBL Filtering. Available: http://www.spamhaus.org/whitepapers/dnsbl_function/. Accessed: Feb. 2014.

[67] Stoica, I., Morris, R., Karger, D., Frans Kaashoek, M., Balakrishnan, H. (Oct. 2001). Chord: A scalable peer-to-peer lookup service for internet applications. ACM SIGCOMM Computer Communications Review - Proceedings of the 2001 SIGCOMM conference, Vol. 31, Issue 4, pp. 149-160.

[68] Stone-Gross, B. (23 Jul. 2012). The Lifecycle of Peer-to-Peer (Gameover) Zeus. Available: http://www.secureworks.com/research/threats/The_Lifecycle_of_Peer_to_Peer_Gameover_ZeuS/. Accessed: Feb. 2014.

[69] Tenable Network Security. (2014). Nessus Vulnerability Scanner. Available: http://www.tenable.com/products/nessus. Accessed: Mar. 2014.

[70] University of Oregon Advanced Network Technology Center. (Jan. 2005). University of Oregon Route Views Project. Available: http://www.routeviews.org/. Accessed: Mar. 2014.

[71] Wikimedia Foundation. Wikipedia:About. Available: http://en.wikipedia.org/wiki/Wikipedia:About. Accessed: Dec. 2013.

[72] Zhou, C.F., Leckie, C., Karunasekera, S. (Feb. 2010). A survey of coordinated attacks and collaborative intrusion detection. Computers and Security, 29 (1), pp. 124-140.