Security Against Probe-Response Attacks in Collaborative Intrusion Detection

Vitaly Shmatikov and Ming-Hsiu Wang
The University of Texas at Austin

ABSTRACT

Probe-response attacks are a new threat for collaborative intrusion detection systems. A probe is an attack which is deliberately crafted so that its target detects and reports it with a recognizable "fingerprint" in the report. The attacker then uses the collaborative infrastructure to learn the detector's location and defensive capabilities from this report.

We analyze the fundamental tradeoff between the ability of a collaborative network to detect epidemic threats and security of individual participants against probe-response attacks. We then design and evaluate a collaborative detection system which provides protection against probe-response attacks. Unlike previously proposed collaborative detection networks, our system supports alert sharing while limiting exposure of members' identities.

1. INTRODUCTION

As network attacks grow in severity and sophistication, *collaborative intrusion detection systems* (CIDS) have attracted much interest. Collecting data from multiple points in the Internet is essential for correlating malicious activity and extracting robust attack signatures. Operational examples include repositories such as DShield [8], DeepSight [22], myNetWatchman [17], IMS [9], and CAIDA [4]. Many architectures have also been proposed for *decentralized* information sharing between Internet monitors [6, 11, 2, 23, 14, 15, 1, 12].

Probe-response attacks are the dark side of collaborative intrusion detection. They turn the usual attack detection game on its head. The attacker *wants* to be detected because he knows that he will be able to use the CIDS as an oracle to analyze the resulting security alert.

Some probe-response attacks aim to map out the regions of the IP address space that are being watched by honeypots and "Internet telescopes," to better avoid them when spreading malware or stage a denial of service attack on them [3, 21]. The attacker probes each address of interest with a special attack whose characteristics are likely to be recognizable in the resulting alert, should the probe be detected. For example, he probes different ports at different target addresses. By analyzing the alerts reported to a collaborative detection center such as DShield, the attacker then learns which probes have been detected and thus which IP addresses are being monitored.

The other type of probe-response attacks, first described in [13], has a somewhat different goal. Consider, for example, a financial institution. Its IP addresses are known and likely to be monitored. The attacker uses the probe-response attack to learn the target's security posture: *how* is a particular type of attack detected, which version of what intrusion detection system is running at a particular IP address, and so on. To the best of our knowledge, the two types of probe-response attacks have not been explicitly differentiated in the research literature, but both present serious risks to the Internet monitors who share information with potentially untrusted peers or potentially vulnerable analysis centers.

Probe-response attacks exploit the paradox of collaborative security. For accurate Internet-scale threat detection, each monitor should reveal his detailed local observations to other monitors. To be safe from probe-response attacks, the monitor should *not* reveal whether he detected an attack. The more useful the reports are for collaborative detection, the more useful they are for an attacker. On the one hand, collaborative detection works best with many diverse monitors. On the other hand, there may not be any observable difference between an honest monitor and a compromised monitor who is quietly leaking his peers' observations to an attacker.

To protect collaborating monitors from probe-response attacks, it is necessary (at the very least) to prevent the attacker from observing *rare* alerts. The probe-response attack in its most basic form exploits the fact that responses to probes have recognizable "fingerprints"—a unique port number, unique Snort rule, and so on—which enable the attacker to recognize them among the alerts collected by the CIDS. If every monitor's observations are reported to the entire community, security against probe-response appears unachievable.

Trusting a central authority to collect and sanitize alerts may not be robust. Even if the authority suppresses rare alerts, it is still the single point of failure: security of the entire system depends on a single node not being compromised. It is also inherently vulnerable to insider attacks.

Decentralized CIDS are potentially more robust, but usually insecure against probe-response attacks. Attacks of the type studied in [3, 21] aim to discover the identities of the collaborating monitors. Most of the previously proposed peer-to-peer detection networks require monitors' identities to be public (to route the alerts), and are thus immediately vulnerable. If a CIDS is to keep the monitors' identities secret, it must solve the following problems: (1) How does a monitor select a peer to send an alert to if peers' identities are secret? (2) How is an alert routed to the selected peer without revealing it to the attacker? and (3) How are the monitors prevented from learning each other's IP addresses?

Of course, suppressing rare alerts impairs the ability of a CIDS to detect low-rate and polymorphic attacks. We argue that there is a fundamental tradeoff between security against probe-response and the ability to detect stealthy attacks, and show how to detect epidemic phenomena while protecting contributors from probe-response attacks.

Our contributions. Our first contribution is a gossip-based architecture for collaborative intrusion detection. It achieves security against probe-response in a fully *decentralized* fashion. The attacker has a very low chance of observing responses to low-density probes (*i.e.*, attacks that affect only a tiny fraction of the monitors), while information about high-density attacks, such as genuine malware outbreaks, is propagated to all monitors. This greatly decreases the efficacy of probe-response attacks by forcing the attacker to stage epidemic-scale probes which yield very imprecise information about the detecting monitors.

Our second contribution is a set of techniques for alert sharing in networks with *hidden membership*. By contrast, previous CIDS assumed that membership is public, revealing exactly the information that probe-response attacks aim to elicit in the first place. Our DHT-based split-route mechanism enables alert distribution while revealing only a small fraction of peers' identities to any given monitor, and without assuming that monitors share cryptographic keys.

Our third contribution is a quantitative analysis of the basic *tradeoff* between a CIDS' ability to detect attacks and security of individual monitors against probe-response. We show that a decentralized CIDS must either sacrifice its ability to detect low-rate threats (including early stages of zero-day attacks), or else expose collaborators to probe-response attacks. This can be viewed as a fundamental limitation of collaborative intrusion detection.

Organization of the paper. We discuss related work in section 2. In section 3, we present an overview of probe-response attacks and the motivation behind our system. The architecture of our system is described in section 4, and the adaptations for public- and hidden-membership networks are in section 5. Our evaluation is presented in section 6. Challenges and future directions appear in section 7.

2. RELATED WORK

Probe-response attacks are first mentioned in [13]; in [3, 21], it is shown how a (somewhat different) probe-response attack can be used to infer the locations of passive Internet monitors. Some countermeasures, *e.g.*, publishing only top lists, sampling, and delayed reporting, are proposed, but they apply to trusted repositories such as DShield [8] rather than distributed detection networks.

Distributed intrusion detection has been the subject of many papers [6, 11, 2, 23, 14, 15, 1, 12, 5]. To enable the monitors to share their observations, these systems assume that they know each other's identities, and thus reveal exactly the information that probe-response attacks aim to discover. In [1], alerts are cryptographically protected, but monitors' locations are not; the system also relies on trusted monitors and a trusted aggregation authority. By contrast, we are interested in a completely *decentralized* detection network that hides both the honest monitors' locations and the contents of their alerts from malicious network members.

DOMINO [23] is superficially similar to our system in that it, too, uses DHT, but its focus is substantially different. CDDHT is a hybrid system that routes different types of alerts in a peer-to-peer fashion to aggregation centers [12].

3. PROBE-RESPONSE ATTACKS

Probe-response attacks target one or a small number of IP addresses. The attack is crafted so that the intrusion alert, if any, generated by the targeted system contains a unique *mark*. For example, an unusual port number can be used as the mark, with different IP addresses attacked on different ports. Port numbers are usually preserved even in anonymized alerts, to enable detection of port scanning sources and correlation of worm instances attacking open network services on a specific port.

In contrast to the usual intrusion detection scenario, the attacker *wants* the attack to be detected. He exploits the basic assumption of collaborative intrusion detection that detected attacks are reported to a public repository or sent to peers. In either case, the alert becomes available to other monitors, some of which are controlled by the attacker. By recognizing the marked alert, the attacker confirms that the targeted IP address is indeed monitored. Detailed analysis of the alert can further reveal how the attack was detected, the make and model of the target's intrusion detection system, and possibly even the target's network topology and enabled network services. In effect, the attacker exploits the detection network to create a feedback loop.

Probe-response attacks were first described in [13]; in [3, 21], it was shown how to exploit DShield and CAIDA alert repositories to learn the locations of "network telescopes" watching unused IP addresses. Probe-response attacks can also aim to discover the detector's capabilities rather than its location. For example, if a given address is watched by an obsolete version of Snort, the protected network may be vulnerable to a recent exploit. The basic pattern—inserting a unique mark into each attack and recognizing it among the collected alerts—remains the same.

Probe-response attacks need not target a small number of addresses. The attacker may launch large-scale attacks (indistinguishable from "genuine" outbreaks), then use statistical analysis of the resulting alerts to find commonalities in their detection patterns. This requires substantially more resources than targeted probes.

Our goal is not to develop a fail-proof defense against every imaginable attack whose aim is to "smoke out" the Internet monitors. Instead, we show the difficulties and tradeoffs in securing collaborative detection networks against basic, narrowly targeted probe-response attacks. Achieving even the *necessary* (let alone sufficient) conditions for security against low-rate probe-response challenges the network's ability to detect "genuine" attacks.

Adversary model. We focus on a *decentralized* intrusion detection network comprising a large number of monitors. The adversary may join the network and/or compromise some fraction of the monitors, but we will assume an upper bound on the number of adversary-controlled monitors. Without such a bound, no defense against probe-response is possible in the absence of a trusted central authority. Defenses against "sybil" attacks [7] are beyond the scope of this paper. In a real-world deployment, the bound on the number of malicious monitors can be achieved, for example, via a pay-to-join scheme with IP address verification.

By assumption, the attacker's observations are limited to the union of the local observations of the monitors he controls. We assume that he cannot eavesdrop on the messages between honest monitors, except those that are routed through an attacker-controlled monitor.

A *probe* is a specially crafted attack on one or more honest monitors. A *response* is the alert generated by an attacked monitor and reported to some of its peers in the detection network. We assume that the mark associated with the probe is recognizable in the response, and the environment is free from noise. Therefore, if any attacker-controlled monitor observes the response, the probe-response attack has succeeded. If anything, these assumptions *strengthen* our attacker. Moreover, it is often possible to filter out the noise by increasing the number of probes [3].

Defenses against probe-response. There is a fundamental conflict at the heart of decentralized intrusion detection. The only incentive for Internet monitors to share information is to obtain access to other monitors' observations (perhaps in an aggregated form). Since the adversary can join the network or corrupt an existing member, we must assume that he has the same access to the data as any other member. Preventing malicious monitors from participating in alert sharing is unrealistic in a truly distributed setting.

To be secure against probe-response attacks, the network must rapidly disseminate reports of genuine attacks to all members, while *not* distributing the alerts resulting from probes that target very few members. (This condition is *necessary*, but not sufficient.) Since a single member, given his local observations, cannot reliably tell the difference between a widespread attack and a probe (the two may look exactly the same, *e.g.*, each involves a malware instance arriving at a certain port), the problem seems impossible.

Our approach is to design an alert propagation mechanism that exploits the difference between the events observed by many monitors and those observed only by a few. For example, the fact that a single monitor somewhere is attacked on port 1434 is of little interest to the entire community.[1] By contrast, if *multiple* monitors are attacked on port 1434, this could be an indication of a worm outbreak, and the corresponding alert should be propagated to as many members as possible to enable pre-emptive defense.

To break the attacker's feedback loop, it is necessary (but not sufficient) to stop him from observing responses to *targeted* probes. The attacker can increase his chances of observing the response by probing multiple addresses with the same attack, but these observations are less useful because the attacker will not know which of the probed IP addresses generated it. This does not rule out probe-response attacks, but at least makes them more expensive.

[1]This is a great simplification. Even a single-host attack may reveal useful data about new vulnerabilities and exploits. Nevertheless, if every host reports all local observations to the entire community, security against probe-response seems unachievable.

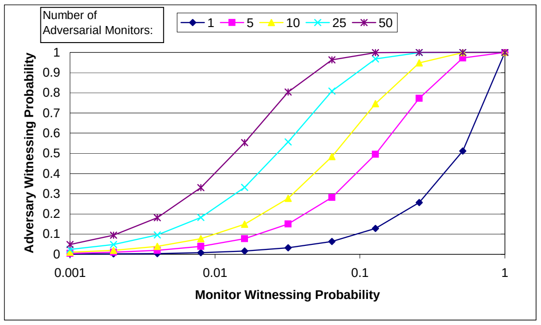


Figure 1: Adversary vs. monitor witnessing probability.

What does security against probe-response mean? Security against probe-response can be characterized by the relationship between attack density α (*i.e.*, the fraction of the monitors attacked using the same probe) and the adversary's "witnessing" probability p_{aw}, *i.e.*, the probability that an attacker-controlled monitor observes the response.

When attack density is low, the witnessing probability should be very low. Alerts corresponding to very rare attacks should "die off" in the network before they reach a non-trivial fraction of monitors. There is no doubt that this decreases effectiveness of collaborative detection, but it is a necessary condition for security against probe-response attacks. We quantify the tradeoff in section 6.

Attacks observed by more than a few of the monitors should be reported to *all* members of the detection network. Let p_{mw} be the probability that a random monitor, whether honest or malicious, witnesses the response. The absolute minimum functionality one should expect from a CIDS is detection of widespread, epidemic threats. For this, high attack densities should be associated with p_{mw} values that are close to 1. Since honest and malicious monitors cannot be reliably distinguished, p_{aw} will be high when p_{mw} is high.

For example, consider

$$p_{aw}(\alpha) = \begin{cases} 0, & \text{if } 0 \leq \alpha < t \\ 1, & \text{if } t \leq \alpha \leq 1 \end{cases}$$

If the CIDS realizes this ideal detection function, all attacks whose density exceeds the *epidemic threshold* t are reported to all members, while reports of lower-density attacks are extinguished. Increasing t achieves better security against probe-response, but also increases the false negative rate, *i.e.*, the probability that a genuine attack is *not* detected by the network. Note that the threshold is *global*: each monitor's local observations may be below t. Achieving this global threshold function in a fully decentralized setting is a non-trivial challenge.

The relationship between p_{aw} (probability that an adversarial monitor witnesses the response) and p_{mw} (probability that a random monitor witnesses the response) is shown in fig. 1. Even though in our alert propagation protocol the chances of two monitors observing an alert are not independent (see section 4), at low attack densities p_{mw} can be approximated as independent for all monitors. Under this assumption, $p_{aw} = 1 - (1 - p_{mw})^A$, where A is the number of adversarial monitors. At higher attack densities, our protocol leads to a stronger dependency between p_{mw} for each monitor m, and this equation overestimates p_{aw}. Note that p_{aw} increases dramatically with small increases in p_{mw}. For the network to be partially secure against even the most basic probe-response attacks, p_{mw} *must* be very low when attack density is low.

Assumptions about the monitors. Since our focus is on the *generic* security of collaborative intrusion detection against probe-response attacks, we abstract away from the actual intrusion detection functionality of each monitor and the exact alert format. It is essential, however, that there exist some similarity metric between the alerts generated by different monitors to enable them to determine whether two or more alerts correspond to similar attacks.

We are aware that such a metric may be problematic for polymorphic attacks. Probe-response attacks, however, fundamentally exploit the fact that the target *does* detect the probe. We expect that there will be at least some commonalities between the monitors' individual observations. For example, with rule-based intrusion detectors, the alert may include the rule that was triggered by the attack along with source IP addresses and port numbers. Typical identifiers for various types of attacks can be found in [12], and techniques for (privacy-preserving) alert matching in [20].

We stress that our goal in this paper is *not* to design a CIDS that can reliably detect zero-day polymorphic attacks. We show that even a much weaker task—detecting conventional attacks, which are observed in exactly the same way by different monitors—conflicts with security against probe-response attacks, and quantify the tradeoff.

4. GOSSIP-BASED CIDS

We now describe the architecture of our collaborative intrusion detection system (CIDS), comprising a network of monitors. For any attack, an honest monitor can be in one of four states: *unaware* (neither detected the attack himself, nor received any alerts about it), *detected* (observed the attack first-hand), *alerted* (has not seen the attack, but has been notified by one or more peers about it), and *confirmed* (learned that the attack is widespread). In our system, a monitor may send an alert to his peers in two situations: alert *generation*, which takes place when the monitor experiences the attack first-hand, and alert *propagation*, when he has received alerts about the attack from his peers.

Alert generation. Upon detecting an attack, the monitor flips n_d biased coins, each of which turns up "heads" with probability p_d. These parameters are, respectively, the *on-detection fanout* and *on-detection alerting probability.* For each "heads," the monitor sends an alert to some random member of the detection network. If members' identities are hidden, the alert is routed as described in section 5. Each alert includes a `pseudoId` field (*e.g.*, a fresh random number), which is unique for each detection event and unlinkable to the monitor's identity. If the monitor observes the same attack more than once, he does not generate a new alert of his own, but may still propagate alerts about this attack received from other members (see below).

Even though we aim to achieve a global threshold for alert dissemination, there is *no local threshold* (unlike in previously proposed CIDS). Even a single attack may result in an alert being sent. With probability $(1-p_d)^{n_d}$, no alerts are sent, thus the absence of an alert no longer indicates that the probe has not been detected.

Alert propagation. When a monitor who has *not* observed the attack first-hand receives an alert about it from a peer monitor, he creates n_r copies of the alert (n_r is the *propagation fanout*). For each copy, he randomly selects a peer and *re-advertises* the alert to him with probability p_r. The value of p_r is a critical parameter. If attack density is low, then p_r should be low, so that alerts corresponding to very rare events die off before reaching an adversarial monitor. If attack density appears to be high, p_r should be high.

Attack density is estimated using `pseudoId`s, which are unique, yet anonymous for each detecting monitor. The number of distinct `pseudoId`s indicates how many distinct monitors observed an instance of a given attack. For each attack, the monitor maintains a list of `pseudoId`s associated with the alerts reporting this attack. When the monitor re-advertises an alert, he picks a `pseudoId` at random from the list. If he previously observed the attack first-hand, he uses the same unique `pseudoId` that he generated then.

Let k be the number of distinct `pseudoId`s. We use different functions for the monitors who observed the attack first-hand ($f_{attacked}$) and those who only heard about it from other monitors ($f_{\sim attacked}$).

The principle behind our choice of functions is as follows. $f_{attacked}$ is exponential to achieve a sharp difference between the (low) probability of a random node learning about a low-density attack and the (high) probability of learning about a high-density attack. $f_{\sim attacked}$ is a threshold function so that (i) if many distinct monitors observed the attack (k is high), then propagation is accelerated; (ii) otherwise, the alert should "die off" in the network.

$$\begin{aligned} f_{attacked}(k) &= \begin{cases} \beta^{(k-\gamma)} & \text{if } 0 \le k \le \gamma \\ 1, & \text{otherwise} \end{cases} \quad (\beta > 1) \\ f_{\sim attacked}(k) &= \begin{cases} 0 & \text{if } 0 \le k \le \tau \\ \rho & \text{otherwise} \end{cases} \quad (\rho < 1) \end{aligned} \qquad (1)$$

In section 6, we explain these functions further, and show how adjusting their parameters affects the tradeoff between security against probe-response and ability to detect attacks.

Because `pseudoId`s are simply long random numbers, the attacker may try to "flush out" the response to his probe by flooding the CIDS with fake `pseudoId`s. This increases honest monitors' estimates of attack density and improves the attacker's probability of observing an alert, but the observed alert is overwhelmingly likely to carry one of the fake `pseudoId`s, yielding no information about the probe's target.

Fig. 2 shows what each monitor does after (i) detecting an attack and (ii) receiving an alert from another monitor.

To establish a more consistent view across the monitors, we set a confirmation threshold c. An honest monitor only considers an attack *confirmed* if he has seen at least c distinct `pseudoId`s for that attack. Regardless of the threshold, even a single response observed by a malicious monitor is sufficient for the probe-response attack to succeed.

Termination of alert propagation. To prevent the infinite cycling of alerts through the network, each monitor can only re-advertise an alert about a given attack k times, where k is the number of distinct `pseudoId`s he has seen in association with this attack. Alerts whose `pseudoId` has already been re-advertised are discarded. Each monitor thus sends at most nK alerts, where $n = \max(n_d, n_r)$, K is the total number of `pseudoId`s, *i.e.*, the number of monitors who observed the attack first-hand. Since $K \le M$, where M is the total number of monitors, the total number of alerts is

```
OnDetectIntrusion (Intrusion i) {
    hasDetect[i] = true;
    pseudoId = random();
    myPseudoIds[i] = pseudoId;
     for 0.. $n_d$ - 1
         with probability $p_d$
             send alert(pseudoId[i], i);
}

OnReceiveAlert (Alert a) {
    if pseudoIdSet[a.i].contains(a.pseudoId)
        return;
     pseudoIdSet[a.i] = pseudoIdSet[a.i] ∪ a.pseudoId;
     k = pseudoIdSet[a.i].size();
     if(hasDetect(a.i)) {
         $p_r$ = $f_{attacked}$(k);
         pseudoId = myPseudoIds[i]
     } else {
         $p_r$ = $f_{\neg attacked}$(k);
        pseudoId = random element in pseudoIdSet[a.i];
     }
      for 0.. $n_r$ -1
         with probability $p_r$
              send alert(pseudoId, i);
}
```

Figure 2: Monitors' algorithms in pseudocode.

bounded by nM^2. To foil an attacker who floods the network with fake `pseudoId`s, monitors can also stop re-advertising the attack after they have seen some fixed number of alerts.

5. HIDING CIDS MEMBERSHIP

We consider two distinct kinds of probe-response attacks. The first, described in [13], assumes that the monitors' locations are known, and aims to reveal their detection capabilities. The second, described in [3, 21], targets detection networks with hidden members, such as Internet telescopes watching "dark" IP addresses. Previously proposed peer-to-peer detection systems assume that members' identities are public, and are thus inherently vulnerable to probe-response attacks of the latter kind.

In a CIDS with public membership, alert propagation is straightforward. A monitor chooses a peer at random (perhaps biasing his selection in favor of trusted peers), and sends the alert via normal IP routing. Standard techniques can be used for confidentiality, authentication, and integrity, provided the monitors share a symmetric key, or the sender has the recipient's certified public key.

In the threat model of [3, 21], however, finding the peers' IP addresses is the main purpose of the attack! This raises a technical challenge: **how to route alerts when peers' locations are not known?** Specifically, (1) How does the monitor select a random peer? (2) How does the monitor send an alert to the selected peer without revealing it to the adversary? and (3) How to prevent the monitors from learning each other's IP addresses?

Selecting a random peer. Following [23, 12], we use a distributed hash table (DHT) to manage CIDS membership. DHT keyspace is typically represented as a circle. Upon joining, each monitor is mapped to a *random* key on this circle, which must be unlinkable to the monitor's true identity. It can be computed, for example, as an HMAC of the monitor's IP address with the monitor's secret key. Each monitor is responsible for some fraction of the DHT keyspace.

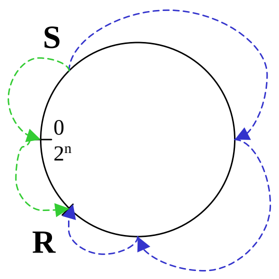


Figure 3: Disjoint routes in DHT keyspace.

To send an alert to a random peer, the monitor picks a random value in the DHT keyspace. The alert is then routed to the monitor responsible for the region of the keyspace containing this value, just like in a normal DHT lookup.

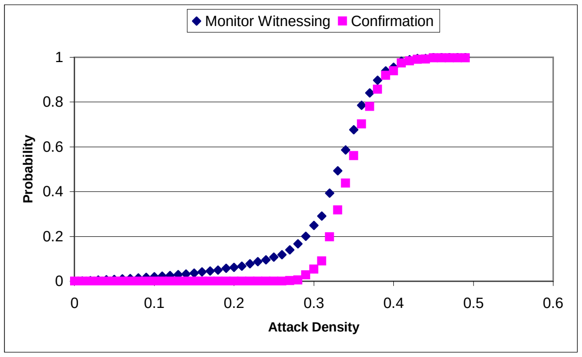
DHT does not guarantee *uniform* peer selection. Each monitor's selection probability is proportional to the range of the keyspace that he is responsible for. For monitors clustered on the circle, this probability is small. Note, however, that to increase his chance of being selected, the adversary must choose a point furthest from all other monitors. This distance is bounded by the largest interval between the keys of the honest monitors. There exist DHT protocols which ensure uniformly random peer selection [10], but in our experiments, non-uniform selection did not lead to a significant increase in the adversary's probability of observing the alert.

Routing alerts. Using DHT means that an alert may transit through several monitors on its way to the destination. If any of them is controlled by the adversary, he will observe the alert and the probe-response attack will succeed. Defense is difficult since the sender and the recipient don't know each other's identity, let alone share cryptographic keys.

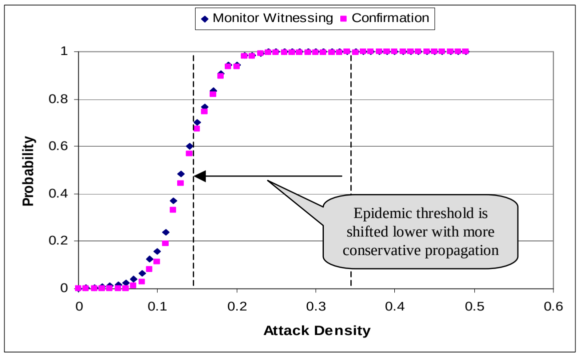
We rely on the *disjoint path defense*. Two routes are disjoint if their only common nodes are the origin and the destination. To route an alert a, the sender generates a random number r of the same length as a, and sends r on one route, and $a \oplus$r on the other, where $\oplus$ is the bitwise `xor`. Upon receiving the two messages, the destination reconstructs the alert by `xor`ing them together. (Optionally, a hash can be included for integrity.) The adversary will not be able to reconstruct the alert from the eavesdropped messages unless *both* routing paths contain an attacker-controlled monitor. This approach easily extends to multiple routes.

Creating disjoint DHT routes. Consider a monitor S sending an alert to a random DHT address x, and R who is responsible for x (see fig. 3). The positions of S and R on the DHT keyspace circle split it into the minor and major arcs, which are clearly disjoint.

In conventional DHT, each intermediate node M routes the message to the neighbor which is closest to R. We modify the protocol by annotating each routing request with either A, or B. M routes messages marked A to the neighbor who lies in the $[M, R]$ interval, and messages marked B



(a) Conservative alert propagation



(b) Aggressive alert propagation

Figure 4: Sample detection curves in a public-membership network.

to the neighbor who lies *outside* the $[M, R]$ interval. In both cases, the distance to R decreases at each hop. Route B is likely to have higher latency, but eventually reaches R.

An alternative protocol is Salsa [18], which is specifically designed to create multiple disjoint routes. Salsa divides the DHT keyspace into groups. Each node knows all members of his group. To create a route, he asks a random set of group members to independently find paths to the destination. The routes created by different group members are only weakly correlated, and disjoint with high probability.

The number of groups is an important parameter. A large number of small groups results in longer routes, increasing the probability that *all* routes contain a malicious node. Moreover, if a monitor finds himself in a group of his own, he cannot create disjoint paths. On the other hand, the smaller each group, the fewer “contacts” each monitor has. In Salsa, with groups of size n, each monitor knows the IP address of every peer in his group, and $\log_2(n)$ peer outside his group. These contacts are the monitors whose IP addresses would be revealed to a dishonest monitor when he joins the CIDS.

6. EVALUATION

To evaluate the CIDS with public membership, we use an event-driven simulator implemented in Java. For the DHT-based CIDS with hidden membership, we use the p2psim [19] simulator for Chord [16] (to simulate standard DHT) and the Java simulator for Salsa [18] (to simulate disjoint-path routing). Each result is an average of 1000 simulations of a network with 500 monitors.

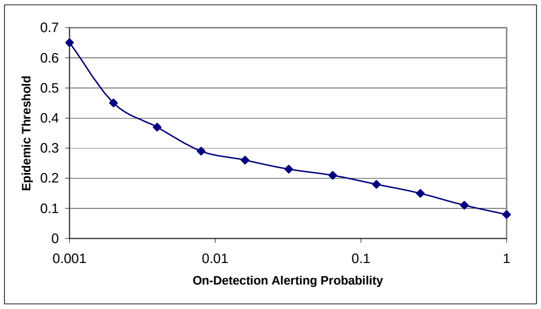


Figure 5: Tuning the epidemic threshold.

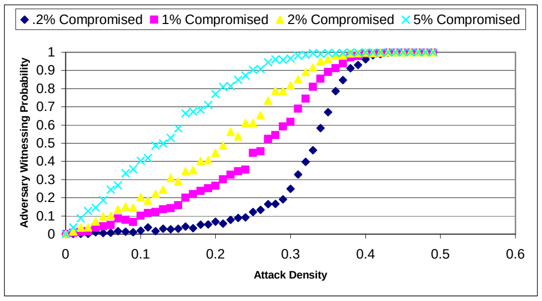


Figure 6: Adversary witnessing probability for a public-membership network.

Public membership. Fig. 4(a) shows the relationship between attack density and the probability of a random monitor learning about the attack, given the on-detection probability $p_d = 0.15$, and parameters of equation (1) set as $\gamma = 5$, $\tau = 4$, $\rho = .7$, and $\beta = 1.23$. (These values are chosen for illustration; below, we explain how adjusting the parameters affects the tradeoffs.) The system is secure against probe-response attacks in the following sense. If few monitors have been attacked, the probability of the attacker observing the response is very small. Only if attack density exceeds 30%, alerts start propagating widely. Fig. 4(a) also shows the effect of attack confirmation (with threshold set to 5). When attack density is between 15% and 30%, multiple monitors see the alerts, but not consider the attack confirmed.

Fig. 4(b) shows the curve for $p_d = 0.25$, $\gamma = 5$, $\tau = 4$, $\rho = .9$, $\beta = 1.01$. With the more aggressive alert propagation function, the epidemic threshold has shifted lower (*i.e.*, lower attack densities are needed to alert all monitors in the network). The tuning of the epidemic threshold by varying p_d is further illustrated in fig. 5.

Fig. 6 shows the attacker's probability of seeing a response to his probe under various assumptions about the fraction of the compromised monitors. Even if the attacker controls *half* of the monitors, his probability of observing the response to a single-target probe is only 7.5%.

The adversary may attack more monitors. If he probes 50 monitors, *i.e.*, 10% of the network, his chance of seeing the response is 2% when .2% of the network is compromised. If he controls 5% of the network, then the probability is 40%. Security against probe-response attacks is hard to achieve when the attacker controls a large fraction of the network. Note, however, that the attack is relatively ineffectual in this case, since the IP address subspace that must be attacked to hit 50 monitors is relatively large, and the attacker learns

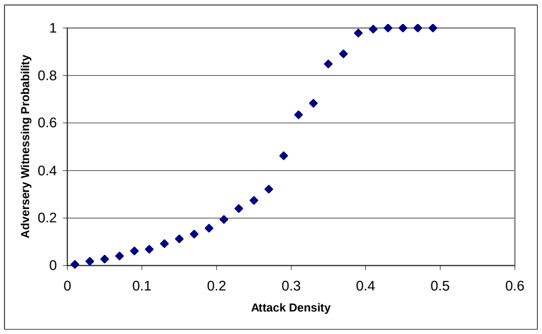


Figure 7: Adversary witnessing probability for Chord with eavesdropping.

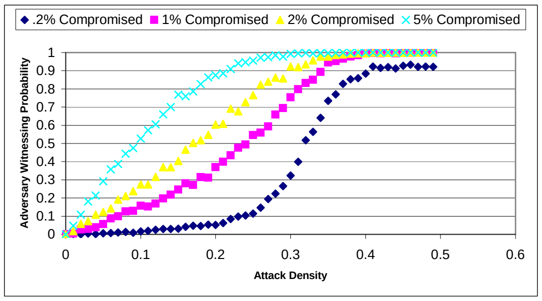


Figure 8: Adversary witnessing probability for Salsa with disjoint path defense.

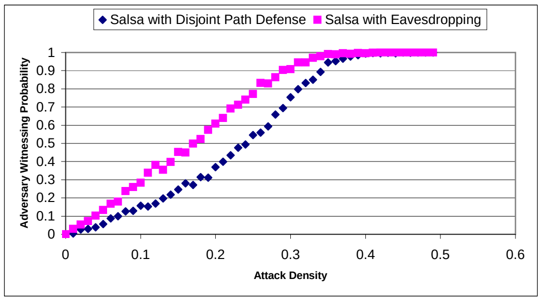


Figure 9: Defending against eavesdropping.

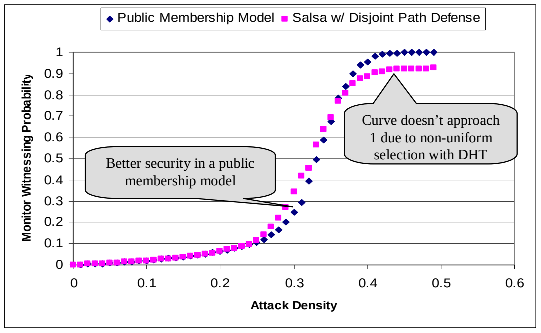


Figure 10: Monitor witnessing probability for a hidden-membership network with Salsa vs. public-membership network.

only that *some* point in this subspace is being monitored.

Hidden membership. Fig. 7 shows the effect of malicious monitors eavesdropping on the alerts routed through, but not addressed to them. Security depends on route length, since longer routes have a higher chance of containing at least one attacker-controlled monitor. These results depend heavily on the specifics of the DHT routing protocol.

Fig. 8 shows the probability that the attacker observes a response to his probe when alerts are routed by Salsa. In our simulations, we used groups with 64 monitors in each; on average, each monitor knows IP addresses of 14 peers. Adversary witnessing probability is slightly worse than in the public-membership network because disjoint paths offer only partial protection against eavesdropping. Effectiveness of the eavesdropping defense is quantified in fig. 9 by comparing the adversary witnessing probability when disjoint paths are used vs. standard DHT routing.

Fig. 10 shows the difference between a public-membership network and a hidden-membership network with Salsa routing. Due to non-uniform peer selection, the curve for the Salsa model does not reach 1. Fig. 10 shows that there is a tradeoff between slightly better security against probe-response attacks in a public-membership network and protection of monitors' identities in a hidden-membership network.

Adjusting parameters. Fig. 11 shows the impact of various parameters of our system on the global detection curve. The lower the on-detection alerting probability and fanout, the flatter the slope of the curve in the low attack density region, which is where probe-response attacks occur.

This is the **fundamental tradeoff** of our system and, we argue, of any decentralized collaborative detection network. Suppressing alert propagation at low attack densities provides better security against probe-response attacks, but also increases the false negative rate, *i.e.*, the probability that a genuine, stealthy attack will not be detected.

The β parameter of $f_{attacked}$ affects the lower bend of the detection curve. As β increases, the curvature of the bend also increases, and the epidemic threshold shifts lower. The γ parameter determines the granularity of the propagation function. Increasing γ alone shifts the epidemic threshold higher as it will take more alerts to obtain the same re-advertisement probability, imposing additional bandwidth and alert processing costs.

For $f_{\sim attacked}$, the τ parameter determines how fast the non-attacked monitors pitch into the gossiping. Ideally, this should occur only if the epidemic threshold has been exceeded. A lower threshold increases the propagation rate, but too low a value may cause premature alert flooding or, even worse, flooding in response to a targeted probe. Similarly, ρ affects the rate of alert propagation once the non-attacked monitors started participating. The effect of τ and ρ is most pronounced in the slope of the detection curve at the epidemic threshold and the curvature of the upper bend.

7. CONCLUSIONS AND FUTURE WORK

We have (1) presented the first characterization of what it means for a collaborative intrusion detection system (CIDS) to be secure against basic probe-response attacks, (2) proposed and evaluated a gossip-based CIDS that achieves our definition of security, and (3) demonstrated the fundamental tradeoff between security against probe-response attacks

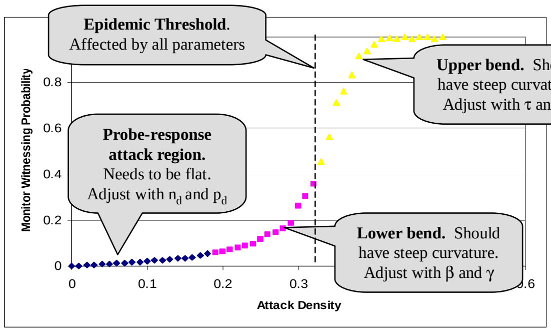


Figure 11: Impact of parameters on the detection curve.

and the system's ability to detect attacks. The main technical contribution is the design of the first decentralized CIDS that supports detection of epidemic attacks while limiting exposure of monitors' identities.

We do not claim to have achieved the Holy Grail of distributed detection of all low-rate, stealthy, polymorphic intrusions, while preventing *any* probe-response attack, including massive attacks with subsequent statistical analysis. Our goal was to show that defending against even the basic probe-response requires compromises. In particular, suppression of reports about low-density attacks appears to be a necessary price to pay, which inevitably decreases the network's ability to detect low-rate threats.

Lifetime of alerts is an important parameter. An adversary may evade detection by waiting until old alerts are purged before attacking the next target. The "memory" of each monitor should be comparable to the attack propagation rate. This is an interesting topic of future research.

Selecting alert destinations uniformly at random may not be the best way to avoid disclosing them to the adversary. In a public-membership network, it may be more secure or efficient to bias selection in favor of trusted or geographically proximate peers.

Other defenses against probe-response attacks may involve deliberately introducing uncertainty into alerts. For example, instead of saying "I observed an intrusion on port 4000," a monitor may say "I observed an intrusion on port 2000, 3000, 4000, or 5000." The techniques of [3] require simultaneous probing of different addresses at different port numbers. Using fake ports in the alerts will provide significant confusion, although it may not be sufficient to defeat a sophisticated attack. If, on the other hand, an epidemic attack is occurring, then a large number of monitors will report the same attack with independent statistical noise in each alert, and standard noise filtering techniques can be used to determine the true port targeted by the attack.

Acknowledgements. This material is based upon work supported by the National Science Foundations under grants CNS-0509033 and IIS-0534198, and by the Department of Defense under Contract No. H98230-05-C-1650.

8. REFERENCES

[1] M. Allman, E. Blanton, V. Paxson, and S. Shenker. Fighting coordinated attackers with cross-organizational information sharing. In *Proc. HotNets*, 2006.

[2] K. Anagnostakis, M. Greenwald, S. Ioannidis, A. Keromytis, and D. Li. A cooperative immunization system for an untrusting Internet. In *Proc. ICON*, 2003.

[3] J. Bethencourt, J. Franklin, and M. Vernon. Mapping Internet sensors with probe response attacks. In *Proc. USENIX Security*, 2005.

[4] CAIDA. `http://www.caida.org`, 2007.

[5] S. Cheetancheri, J. Agosta, D. Dash, K. Levitt, J. Rowe, and E. Schooler. A distributed host-based worm detection system. In *Proc. LSAD*, 2006.

[6] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *Proc. RAID*, 2001.

[7] J. Douceur. The Sybil attack. In *Proc. IPTPS*, 2002.

[8] DShield. `http://www.dshield.org`, 2006.

[9] Internet Motion Sensor. `http://ims.eecs.umich.edu/`, 2006.

[10] V. King, S. Lewis, and J. Saia. On algorithms for choosing a random peer. Technical Report TR-CS-2004-31, Dept. of Computer Science, University of New Mexico, 2005.

[11] C. Krügel, T. Toth, and C. Kerer. Decentralized event correlation for intrusion detection. In *Proc. ICISC*, 2001.

[12] Z. Li, Y. Chen, and A. Beach. Towards scalable and robust distributed intrusion alert fusion with good load balancing. In *Proc. LSAD*, 2006.

[13] P. Lincoln, P. Porras, and V. Shmatikov. Privacy-preserving sharing and correlation of security alerts. In *Proc. USENIX Security*, 2004.

[14] M. Locasto, J. Parekh, A. Keromytis, and S. Stolfo. Towards collaborative security and P2P intrusion detection. In *Proc. IAW*, 2005.

[15] D. Malan and M. Smith. Host-based detection of worms through peer-to-peer cooperation. In *Proc. WORM*, 2005.

[16] R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. SIGCOMM*, 2001.

[17] myNetWatchman. `http://www.mynetwatchman.com`, 2006.

[18] A. Nambiar and M. Wright. Salsa: a structured approach to large-scale anonymity. In *Proc. CCS*, 2006.

[19] p2psim. `http://pdos.csail.mit.edu/p2psim/`, 2005.

[20] J. Parekh, K. Wang, and S. Stolfo. Privacy-preserving payload-based correlation for accurate malicious traffic detection. In *Proc. LSAD*, 2006.

[21] Y. Shinoda, K. Ikai, and M. Itoh. Vulnerabilities of passive Internet threat monitors. In *Proc. USENIX Security*, 2005.

[22] Symantec. DeepSight threat management system. `http://tms.symantec.com`, 2006.

[23] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the DOMINO overlay system. In *Proc. NDSS*, 2004.