Fast Detection of Scanning Worm Infections

Stuart E. Schechter[1], Jaeyeon Jung[2], and Arthur W. Berger[2]

[1] Harvard DEAS,
33 Oxford Street, Cambridge MA 02138, USA,
stuart@eecs.harvard.edu
[2] MIT CSAIL,
32 Vassar Street, Cambridge MA 02139, USA,
{jyjung,awberger}@csail.mit.edu

Abstract. Worm detection and response systems must act quickly to identify and quarantine scanning worms, as when left unchecked such worms have been able to infect the majority of vulnerable hosts on the Internet in a matter of minutes [9]. We present a hybrid approach to detecting scanning worms that integrates significant improvements we have made to two existing techniques: sequential hypothesis testing and connection rate limiting. Our results show that this two-pronged approach successfully restricts the number of scans that a worm can complete, is highly effective, and has a low false alarm rate.

1 Introduction

Human reaction times are inadequate for detecting and responding to fast scanning worms, such as `Slammer`, which can infect the majority of vulnerable systems in a matter of minutes [18, 9]. Thus, today's worm response proposals focus on *automated* responses to worms, such as quarantining infected machines [10], automatic generation and installation of patches [14, 15], and reducing the rate at which worms can issue connection requests so that a more carefully constructed response can be crafted [22, 27].

Even an automated response will be of little use if it fails to be triggered quickly after a host is infected. Infected hosts with high-bandwidth network connections can initiate thousands of connection requests per second, each of which has the potential to spread the infection. On the other hand, an automated response that triggers too easily will erroneously identify hosts as infected, interfering with these hosts' reliable performance and causing significant damage.

Many scan detection mechanisms rely upon the observation that only a small fraction of addresses are likely to respond to a connection request at any given port. Many IPv4 addresses are dead ends as they are not assigned to active hosts. Others are assigned to hosts behind firewalls that block the port addressed by the scanner. When connection requests do reach active hosts, many will be rejected as not all hosts will be running the targeted service. Thus, scanners are likely to have a low rate of successful connections, whereas benign hosts, which only issue connection requests when there is reason to believe that addressees will respond, will have a much greater rate of success.

Existing methods for detecting scanning worms within a local network use fixed thresholds for the number of allowable failed connections over a time period [16] or limit the rate at which a host can initiate contact with additional hosts [27]. However, these threshold based approaches may fail to detect low-rate scanning. They may also require an excessive number of connection observations to detect an infection or lead to an unnecessary number of false alarms.

To detect inbound scans initiated by hosts outside the local network, previous work on which we collaborated [7] used an approach based on sequential hypothesis testing. This approach automatically adjusts the number of observations required to detect a scan with the strength of the evidence supporting the hypothesis that the observed host is, in fact, scanning. The advantage of this approach is that it can reduce the number of connection requests that must be observed to detect that a remote host is scanning while maintaining an acceptable false alarm rate.

While this approach shows promise for quickly detecting scanning by hosts inside a local network soon after they have been infected by a worm, there are significant hurdles to overcome. For one, to determine whether a request to connect to a remote host will fail, one must often wait to see whether a successful connection response will be returned. Until enough connection requests can be established to be failures, a sequential hypothesis test will lack the observations required to conclude that the system is infected. By the time the decision to quarantine the host is made, a worm with a high scan rate may have already targeted thousands of other hosts.

This earlier work used a single sequential hypothesis test per host and did not re-evaluate benign hosts over time. Unlike an intrusion detection system observing remote hosts, a worm detection system is likely to observe benign traffic originating from an infected host before it is infected. It is therefore necessary to adapt this method to continuously monitor hosts for indications of scanning.

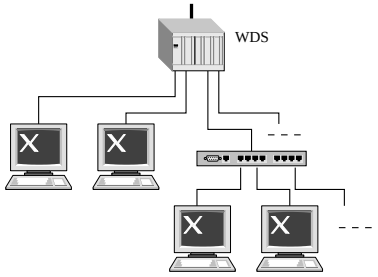

Fig. 1. A Worm Detection System (WDS) is located to monitor a local network

We introduce an innovative approach that enables a Worm Detection System (WDS) to continuously monitor a set of *local* hosts for infection, requiring a small number of observations to be collected after an infection to detect that the host is scanning (Figure 1).

To detect infected hosts, the WDS need only process a small fraction of network events; a subset of connection request observations that we call *first-contact connection* requests and the responses to these requests that complete the connections. A first-contact connection request is a packet (TCP or UDP) addressed to a host with which the sender has not previously communicated. These events are monitored because scans are mostly composed of first-contact connection requests.

In Section 2, we introduce a scan detection algorithm that we call a reverse sequential hypothesis test ($\overleftarrow{HT}$), and show how it can reduce the number of first-contact connections that must be observed to detect scanning[3]. Unlike previous methods, the number of observations $\overleftarrow{HT}$ requires to detect hosts' scanning behavior is not affected by the presence of benign network activity that may be observed before scanning begins.

In Section 3, we introduce a new credit-based algorithm for limiting the rate at which a host may issue the first-contact connections that are indicative of scanning activity. This credit-based connection rate limiting (CBCRL) algorithm results in significantly fewer false positives (unnecessary rate limiting) than existing approaches.

When combined, this two-pronged approach is effective because these two algorithms are complementary. Without credit-based connection rate limiting, a worm could rapidly issue thousands of connection requests before enough connection failures have been observed by Reverse Sequential Hypothesis Testing so that it can report the worm's presence. Because Reverse Sequential Hypothesis Testing processes connection success and failure events in the order that connection requests are issued, false alarms are less likely to occur than if we used an approach purely based on credit-based connection rate limiting, for which first-contact connections attempts are assumed to fail until the evidence proves otherwise.

We demonstrate the utility of these combined algorithms with trace-driven simulations, described in Section 4, with results presented in Section 5. The limitations of our approach, including strategies that worms could attempt to avoid detection, are presented in Section 6. We discuss related work, including previous approaches to the scanning worm detection problem, in Section 7. Our plans for future work are presented in Section 8, and we conclude in Section 9.

2 Detecting Scanning Worms by Using Reverse Sequential Hypothesis Testing

A worm is a form of malware that spreads from host to host without human intervention. A scanning worm locates vulnerable hosts by generating a list of addresses to probe and then contacting them. This address list may be generated sequentially or pseudo-randomly. Local addresses are often preferentially

[3] The letters in this abbreviation, $\overleftarrow{HT}$, stand for Hypothesis Testing and the arrow indicates the reverse sequential order in which observations are processed.

selected [25] as communication between neighboring hosts will likely encounter fewer defenses. Scans may take the form of TCP connection requests (SYN packets) or UDP packets. In the case of the connectionless UDP protocol, it is possible for the scanning packet to also contain the body of the worm as was the case with the `Slammer` worm [9].

In this section, we present an on-line algorithm for detecting the presence of scanners within a local network by observing network traffic. We use a sequential hypothesis test for its ability to adjust the number of observations required to make a decision to match the strength of the evidence it is presented with.

2.1 Sequential Hypothesis Testing

As with existing approaches to scan detection [7, 17, 22, 27], we rely upon the observation that only a small fraction of addresses are likely to respond to a connection request at any given port. Benign hosts, which only contact systems when they have reason to believe that this connection request will be accepted, are more likely to receive a response to a connection request.

Recall that a first-contact connection request is a packet (TCP or UDP) addressed to a host with which the sender has not previously communicated. When a local host l initiates a first-contact connection request to a destination address, d, we classify the outcome as either a "success" or a "failure". If the request was a TCP SYN packet, the connection is said to succeed if a SYN-ACK is received from d before a timeout expires. If the request is a UDP packet, any UDP packet from d received before the timeout will do. We let Y_i be a random (indicator) variable that represents the outcome of the i^{th} first-contact connection request by l, where

$$Y_i = \begin{cases} 0 & \text{if the connection succeeds} \\ 1 & \text{if the connection fails} \end{cases}$$

Detecting scanning by local hosts is a problem that is well suited for the method of *sequential hypothesis testing* first developed by Wald [24], and used in our earlier work to detect remote scanners [7].

We call H_1 the hypothesis that host l is engaged in scanning (indicating infection by a worm) and H_0 the null hypothesis that the host is not scanning. We assume that, conditional on the hypothesis H_j, the random variables $Y_i|H_j$ $i = 1, 2, \ldots$ are independent and identically distributed (i.i.d.). That is, conditional on the hypothesis, any two connection attempts will have the same likelihood of succeeding, and their chances of success are unrelated to each other. We can express the distribution of the Bernoulli random variable Y_i as:

$$\begin{aligned} \Pr[Y_i = 0|H_0] &= \theta_0, \quad & \Pr[Y_i = 1|H_0] &= 1 - \theta_0 \\ \Pr[Y_i = 0|H_1] &= \theta_1, \quad & \Pr[Y_i = 1|H_1] &= 1 - \theta_1 \end{aligned}$$

Given that connections originating at benign hosts are more likely to succeed than those initiated by a scanner, $\theta_0 > \theta_1$.

Sequential hypothesis testing chooses between two hypotheses by comparing the likelihoods that the model would generate the observed sequence of events, $\mathbf{Y}_n \equiv (Y_1, \ldots, Y_n)$, under each hypothesis. It does this by maintaining the ratio $\Lambda(\mathbf{Y}_n)$, the numerator of which is the likelihood that the model would generate the sequence of events $\mathbf{Y}_n$ under hypothesis H_1, and the denominator under hypothesis H_0.

$$\Lambda(\mathbf{Y}_n) \equiv \frac{\Pr[\mathbf{Y}_n|H_1]}{\Pr[\mathbf{Y}_n|H_0]} \tag{1}$$

The i.i.d. assumption in the model enables us to state this ratio in terms of the likelihoods of the individual events.

$$\Lambda(\mathbf{Y}_n) \equiv \prod_{i=1}^{n} \frac{\Pr[Y_i|H_1]}{\Pr[Y_i|H_0]} \tag{2}$$

We can write the change to $\Lambda(\mathbf{Y}_n)$ as a result of the i^{th}observation as $\phi(Y_i)$:

$$\phi(Y_i) \equiv \frac{\Pr[Y_i|H_1]}{\Pr[Y_i|H_0]} = \begin{cases} \frac{\theta_1}{\theta_0} & \text{if } Y_i = 0 \text{ (success)} \\ \\ \frac{1-\theta_1}{1-\theta_0} & \text{if } Y_i = 1 \text{ (failure)} \end{cases}$$

This enables us to rewrite $\Lambda(\mathbf{Y}_n)$ inductively, such that $\Lambda(\mathbf{Y}_0) = 1$, and $\Lambda(\mathbf{Y}_n)$ may be calculated iteratively as each observation arrives.

$$\Lambda(\mathbf{Y}_n) = \prod_{i=1}^{n} \phi(Y_i) = \Lambda(\mathbf{Y}_{n-1})\phi(Y_n)$$

One compares the likelihood ratio $\Lambda(\mathbf{Y}_n)$ to an upper threshold, η_1, above which we accept hypothesis H_1, and a lower threshold, η_0, below which we accept hypothesis H_0. If $\eta_0 < \Lambda(\mathbf{Y}_n) < \eta_1$ then the result will remain inconclusive until more events in the sequence can be evaluated. This is illustrated in Figure 2.

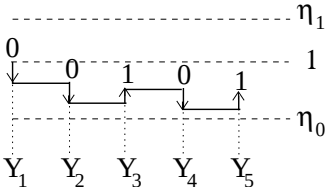

Fig. 2. A log scale graph of $\Lambda(\mathbf{Y})$ as each observation, Y_i, is added to the sequence. Each success (0) observation decreases $\Lambda(\mathbf{Y})$, moving it closer to the benign conclusion threshold η_0, whereas each failure (1) observation increases $\Lambda(\mathbf{Y})$, moving it closer to the infection conclusion threshold η_1

Writing the probability of correctly reporting detection (declaring host is infected when indeed it is) as P_D and the probability of a false positive (declaring

host is infected when in fact it is not) as P_F, we can define our performance requirements as bounds α and β on these probabilities.

$$\alpha \geq P_F \quad \text{and} \quad \beta \leq P_D$$

Because every false positive can decrease productivity of both the users of a host and the security staff who need to inspect it, one would expect to use α values that are small fractions of a percentage point. Since scanners generate enough traffic to clearly differentiate their behavior from that of benign systems, a β of greater than 0.99 should be an achievable requirement.

Wald [24] showed that η_1 and η_0 can be bounded in terms of P_D and P_F.

$$\eta_1 \leq \frac{P_D}{P_F} \tag{3}$$

$$\frac{1 - P_D}{1 - P_F} \leq \eta_0 \tag{4}$$

Given our requirement parameters α and β, we assign the following values to our thresholds, η_0 and η_1:

$$\eta_1 \leftarrow \frac{\beta}{\alpha} \tag{5}$$

$$\eta_0 \leftarrow \frac{1 - \beta}{1 - \alpha} \tag{6}$$

From Equations (3) and (5), we can bound P_F in terms of α and β. Since $0 < P_D < 1$, we can replace P_D with 1 in Equation (3) to yield:

$$\eta_1 \leq \frac{P_D}{P_F} < \frac{1}{P_F} \tag{7}$$

It follows that:

$$P_F < \frac{1}{\eta_1} = \frac{\alpha}{\beta} \tag{8}$$

Likewise, using Equation (4) and given that $1 - P_D < (1 - P_D)/(1 - P_F)$, we can bound $1 - P_D$:

$$1 - P_D < \eta_0 = \frac{1 - \beta}{1 - \alpha} \tag{9}$$

While η_1 may result in a false positive rate above our desired bound by a factor of $\frac{1}{\beta}$, this difference is negligible given our use of β values in the range of 0.99 and above. Similarly, while our miss rate, $1 - P_D$ may be off by as much as a factor of $\frac{1}{1-\alpha}$, this too will have negligible effect given our requirements for very small values of α.

2.2 Detecting Infection Events

In our earlier work, it was assumed that each *remote* host was either a scanner or benign for the duration of the observed period. When a host was determined to be benign it would no longer be observed. In contrast, in this paper we are concerned with detecting infection events, in which a *local* host transitions from a benign state to an infected state. Should a host become infected while a hypothesis test is already running, the set of outcomes observed by the sequential hypothesis test may include those from both the benign and infected states, as shown in Figure 3. Even if we continue to observe the host and start a new hypothesis test each time a benign conclusion is reached, the test may take longer than necessary to conclude that an infection has occurred.

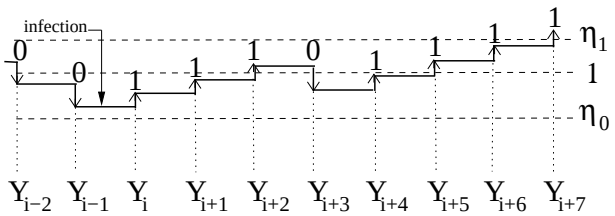

Fig. 3. A log scale graph tracing the value of $\Lambda(\mathbf{Y})$ as it is updated for a series of observations that includes first-contact connection requests before (Y_{i-1} and Y_{i-2}) and after (Y_i and beyond) the host was infected

The solution to this problem is to run a new sequential hypothesis test as each connection outcome is observed, evaluating these outcomes in reverse chronological order, as illustrated in Figure 4. To detect a host that was infected before it issued first-contact connection i (event Y_i), but after it had issued first-contact connection $i-1$, a reverse sequential hypothesis test ($\overleftarrow{HT}$) would require the same number of observations to detect the infection as would a forward sequential hypothesis that had started observing the sequence at observation i. Because the most recent observations are processed first, the reverse test will terminate before reaching the observations that were collected before infection.

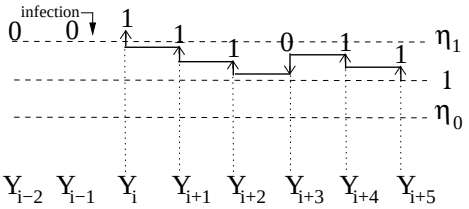

Fig. 4. A log scale graph tracing the value of $\Lambda(Y_{i+5}, Y_{i+4}, \ldots)$, in which the observations in $\mathbf{Y}$ are processed in *reverse* sequential order. The most recent, or rightmost, observation is the first one processed

When we used sequential hypothesis testing in our prior work to detect scanning of a local network by remote hosts, the intrusion detection system could know *a priori* whether a connection would fail given its knowledge of the network topology and services [7]. Thus, the outcome of a connection request from host i could immediately be classified as a success or failure observation (Y_i) and $\Lambda(\mathbf{Y}_n)$ could be evaluated without delay.

When a *local* host initiates first-contact connection requests to remote hosts, such as those shown in Figure 5, the worm detection system cannot immediately determine if the connection will succeed or fail. While some connection failures will result in a TCP RST packet or an ICMP packet [1, 3], empirical evidence has shown that most do not [2]. The remaining connection attempts can be classified as failures only after a timeout expires.

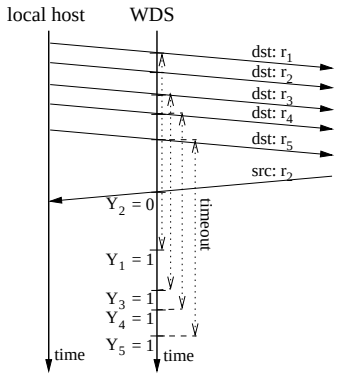

Fig. 5. The success of first-contact connection requests by a local host to remote hosts cannot be established by the Worm Detection System (WDS) until a response is observed or a timeout expires

While a sequential hypothesis test waits for unsuccessful connections to time out, a worm may send thousands of additional connection requests with which to infect other systems. To limit the number of outgoing first-contact connections, a sequential hypothesis testing approach can be paired with a credit-based connection rate limiter as described in Section 3.

2.3 Algorithmic Implementation

A naïve implementation of repeated reverse sequential hypothesis testing requires that we store an arbitrarily large sequence of first-contact connection observations. A naïve implementation must also step through a portion of this sequence each time a new observation is received in order to run a new test starting at that observation.

Fortunately, there exists an iterative function:

$$\bar{\Lambda}(\mathbf{Y}_n) = \max\left(1,\ \bar{\Lambda}(\mathbf{Y}_{n-1})\phi(Y_n)\right)$$

with state variable $\bar{\Lambda}(\mathbf{Y}_n)$, that can be calculated in the sequence in which events are observed, and that has the property that its value will exceed η_1 if and only if a reverse sequential hypothesis test would conclude from this sequence that the host was infected. This is proved in Appendix A.

Updating $\bar{\Lambda}$ for each observation requires only a single multiplication and two comparison operations[4]. Because $\bar{\Lambda}$ is updated in sequence, observations can be discarded immediately after they are used to update the value of $\bar{\Lambda}$.

When running this algorithm in a worm detection system, we must maintain separate state information for each host being monitored. Thus, a state variable $\bar{\Lambda}_l$ is maintained for each local host l.

It is also necessary to track which hosts have been previously contacted by l. We track the set of Previously Contacted Hosts, or `PCH` set, for each local host.

```
enum status {PENDING, SUCCESS, FAILURE};
struct FCC_Queue_Entry {
  ip4_addr DestAddr;
  time     WhenInitiated;
  status   Status;
}
```

Fig. 6. The structure of entries in the First-Contact Connection (FCC) queue

Finally, each local host l has an associated queue of the first-contact connection attempts that l has issued but that have not yet been processed as observations. The structure of the records that are pushed on this `FCC` queue are shown in Figure 6. The choice of a queue for this data structure ensures that first-contact connection attempts are processed in the order in which they are issued, not in the order in which their status is determined.

The algorithm itself is quite simple and is triggered upon one of three events.

1. When the worm detection system observes a packet (TCP SYN or UDP) sent by local host l, it checks to see if the destination address d is in l's previously contacted host (`PCH`) set. If it isn't, it adds d to the `PCH` set and adds a new entry to the end of the `FCC` queue with d as the destination address and status `PENDING`.
2. When an incoming packet arrives addressed to local host l and the source address is also the destination address (`DestAddr`) of a record in l's `FCC` queue, the packet is interpreted as a response to the first-contact connection request and the status of the `FCC` record is updated. The status of the `FCC` record is set to `SUCCESS` unless the packet is a TCP RST packet, which indicates a rejected connection.
3. Whenever the entry on the front of the `FCC` queue has status `PENDING` and has been in the queue longer than the connection timeout period, a timeout occurs and the entry is assigned the status of `FAILURE`.

[4] In fact, addition and subtraction operations are adequate as the iterative function is equivalent to $\Theta(\mathbf{Y}_n) = \max(0,\ \Theta(\mathbf{Y}_{n-1}) + \ln\phi(Y_n))$ where $\Theta(\mathbf{Y}_n) \equiv \ln\bar{\Lambda}(\mathbf{Y}_n)$.

When any of the above events causes the entry at the front of the FCC queue to have status other than PENDING, it is dequeued and $\bar{\Lambda}_l$ is updated and compared to η_1. If $\bar{\Lambda}_l \geq \eta_1$, we halt testing for host l and immediately conclude that l is infected. Dequeuing continues so long as $\bar{\Lambda}_l < \eta_1$, the front entry of the FCC queue has status other than PENDING, and the queue is not empty.

3 Slowing Worm Propagation by Using Credit-Based Connection Rate Limiting

It is necessary to limit the rate at which first-contact connections can be initiated in order to ensure that worms cannot propagate rapidly between the moment scanning begins and the time at which the scan's first-contact connections have timed out and been observed by our reverse sequential hypothesis test ($\overleftarrow{HT}$).

Twycross and Williamson [27, 22] use a technique they call a virus throttle to limit outgoing first-contact connections. When observing a given host, their algorithm maintains a working set of up to five hosts previously contacted by the host they are observing. For the purpose of their work, a first-contact connection is a connection to a host not in this working set. First-contact connections issued when the working set is full are not sent out, but instead added to a queue. Once per second the least recently used entry in the working set is removed and, if the pending queue of first-contact connection requests is not empty, a request is pulled off the queue, delivered, and its destination address is added to the working set. All requests in the queue with the same destination address are also removed from the queue and delivered.

Virus throttling is likely to interfere with HTTP connection requests for inlined images, as many Web pages contain ten or more inlined images each of which is located on a distinct peering server. While a slow but bursty stream of requests from a Web browser will eventually be released by the throttle, mail servers, Web crawlers, and other legitimate services that issue first-contact connections at a rate greater than once per second will overflow the queue. In this case, the virus throttling algorithm quarantines the host and allows no further first-contact connections.

To achieve rate limiting with a better false positive rate we once again present a solution inspired by sequential hypothesis testing and that relies on the observation that benign first-contact connections are likely to succeed whereas those issued by scanners are likely to fail. This credit-based approach, however, is unlike $\overleftarrow{HT}$ in that it assumes that a connection will fail until evidence proves otherwise. Because it does not wait for a timeouts to act, it can react immediately to a burst of connections and halt the flow so that $\overleftarrow{HT}$ can then make a more informed decision as to whether the host is infected. As it does not force connections to be evaluated in order, CBCRL can also immediately process evidence of connection successes. This will enable it to quickly increase the allowed first-contact connection rate when these requests are benign.

Credit-based connection rate limiting, as summarized in Figure 7, works by allocating to each local host, l, a starting balance of ten credits ($C_l \leftarrow 10$)

which can be used for issuing first-contact connection requests. Whenever a first-contact connection request is observed, a credit is subtracted from the sending host's balance ($C_l \leftarrow C_l - 1$). If the successful acknowledgment of a first-contact connection is observed, the host that initiated the request is issued two additional credits ($C_l \leftarrow C_l + 2$). No action is taken when connections fail, as the cost of issuing a first-contact connection has already been deducted from the issuing host's balance. Finally, first-contact connection requests are blocked if the host does not have any credit available ($C_l = 0$)[5].

Event	Change to C_l
Starting balance	$C_l \leftarrow 10$
FCC issued by l	$C_l \leftarrow C_l - 1$
FCC succeeds	$C_l \leftarrow C_l + 2$
Every second	$C_l \leftarrow \max(10, \frac{2}{3}C_l)$ if $C_l > 10$
Allowance	$C_l \leftarrow 1$ if $C_l = 0$ for 4 seconds

Fig. 7. The underlying equations behind credit-based connection rate limiting. Changes to a host's balance are triggered by the first-contact connections (FCCs) it initiates and by the passing of time

If a first-contact connection succeeds with probability θ, its expected payoff from issuing that connection is its expected success credit minus its cost, or $2\theta - 1$. This payoff is positive for $\theta > \frac{1}{2}$ and negative otherwise. Hosts that scan with a low rate of successful connections will quickly consume their credits whereas benign hosts that issue first-contact connections with high rates of success will nearly double their credits each time they invest them.

As described so far, the algorithm could result in two undesirable states. First, a host could acquire a large number of credits while performing a benign activity (e.g. Web crawling) which could be used later by a scanning worm. Second, a network outage could cause a benign host to use all of its credits after which it would starve for a lack of first-contact connection successes.

These problems are addressed by providing each host with a small allowance and by putting in place a high rate of inflation. If a host has been without credits for four seconds, we issue the host a single credit ($C_l \leftarrow 1$ if $C_l \leq 0$). This not only ensures that the host does not starve, but enables us to collect another observation to feed into our hypothesis test ($\overleftarrow{HT}$). Because $\overleftarrow{HT}$, as configured in Section 4, observes all first-contact connection requests as successes or failures within three seconds, providing a starving process with a credit allowance only after more than three seconds have passed ensures that $\overleftarrow{HT}$ will have been executed on all previously issued first-contact connection requests. If $\overleftarrow{HT}$ has already concluded that the host is a worm, it is expected that the system will be quarantined and so no requests will reach their destination regardless of the credit balance.

[5] In Section 8, we discuss the alternative of allowing all TCP requests to be transmitted and queueing responses until credits are available.

For each second that passes, a host that has acquired more than 10 credits will be forced to surrender up to a third of them, but not so many as to take its balance below 10 ($C_l \leftarrow \max(10, \frac{2}{3}C_l)$ if $C_l > 10$). A host that is subject to the maximum inflation rate, with a first-contact connection rate r, success rate $\theta > 0$, and credit balance $C_{l,t}$ at time t, will see this balance reach an equilibrium state $\hat{C}$ when $\hat{C} = C_{l,t} = C_{l,t+1}$.

$$\begin{aligned} C_{l,t+1} &= \frac{2}{3}(C_{l,t} + r \cdot (2\theta - 1)) \\ \hat{C} &= \frac{2}{3}(\hat{C} + r \cdot (2\theta - 1)) \\ \hat{C} &= \frac{2}{3}\hat{C} + \frac{2}{3} \cdot r \cdot (2\theta - 1) \\ \frac{1}{3}\hat{C} &= \frac{2}{3} \cdot r \cdot (2\theta - 1) \\ \hat{C} &= 2 \cdot r \cdot (2\theta - 1) \end{aligned}$$

One can now see that we chose the inflation constant $\frac{2}{3}$ to ensure that, in the upcoming second, a host that has a perfect first-contact connection success rate ($\theta = 1$) will have twice as many credits as it could have needed in the previous second. Also note that the maximum inflation rate, which seems quite steep, is only fully applied when $\hat{C} \geq 15$, which in turn occurs only when the first-contact connection rate r is greater than 7.5 requests per second. Twycross and Williamson's virus throttle, on the other hand, can only assume that any host with a first-contact connection rate consistently greater than one request per second is a worm.

The constant of 10 was chosen for the starting credit balance (and for the equilibrium minimum credit balance for benign hosts with first-contact connection rates below 5 requests/second) in order to match the requirements of our sequential hypothesis test ($\overleftarrow{HT}$) as currently configured (see parameters in Section 4), which itself requires a minimum of 10 observations in order to conclude that a host is engaged in scanning. Slowing the rate at which the first 10 observations can be obtained will only delay the time required by $\overleftarrow{HT}$ to conclude that a host is engaged in scanning. Should the parameters of $\overleftarrow{HT}$ be reconfigured and the minimum number of observations required to conclude a host is a scanner change, the starting credit balance for rate-limiting can be changed to match it.

4 Experimental Setup

We evaluated our algorithms using two traces collected at the peering link of a medium sized ISP; one collected in April 2003 (`isp-03`) containing 404 active hosts and the other in January 2004 (`isp-04`) containing 451 active hosts. These traces, summarized in Table 1, were collected using `tcpdump`.

Obtaining usable traces was quite difficult. Due to privacy concerns, network administrators are particularly loathe to share traces, let alone those that contain

payload data in addition to headers. Yet, we required the payload data in order to manually determine which, if any, worm was present on a host that was flagged as infected.

Table 1. Summary of network traces

	`isp-03`	`isp-04`
Date	2003/04/10	2004/01/28
Duration	627 minutes	66 minutes
Total outbound connection attempts	1,402,178	178,518
Total active local host	404	451

To best simulate use of our algorithm in a worm detection system that is used to quarantine hosts, we only tested local hosts for infection. Remote hosts were not tested.

In configuring our reverse sequential hypothesis test ($\overleftarrow{HT}$), first-contact connection requests were interpreted as failures if they were not acknowledged within a three second grace period. First-contact connection requests for which TCP RST packets were received in response were immediately reported as failure observations. Connection success probability estimates were chosen to be:

$$\theta_0 = 0.7 \qquad \theta_1 = 0.1$$

Confidence requirements were set to:

$$\alpha = 0.00005 \qquad \beta = 0.99$$

Note that these confidence requirements are for each reverse sequential hypothesis test, and that a test is performed for each first-contact connection that is observed. Therefore, the false positive rate is chosen to be particularly low as testing will occur many times for each host.

For each local host we maintained a Previously Contacted Host (`PCH`) set of only the last 64 destination addresses that each local host had communicated with (LRU replacement). For the sake of the experiment, a first-contact connection request was any TCP SYN packet or UDP packet addressed to a host that was not in the local host's `PCH` set. While using a fixed sized `PCH` set demonstrates the efficacy of our test under the memory constraints that are likely to occur when observing large (e.g. class B) networks, this fixed memory usage comes at a cost. As described in Section 6, it is possible for a worm to exploit limitations in the `PCH` set size in order to avoid having its scans detected.

For sake of comparison, we also implemented Twycross and Williamson's 'virus throttle' as described in [22]. Since our traces contain only those packets seen at the peering point, our results may differ from a virus throttle implemented at each local host as Twycross and Williamson recommend. However, because observing connections farther from the host results in a reduction in

the number of connections observed, it should only act to reduce the reported number of false positives in which benign behavior is throttled.

All algorithms were implemented in `Perl`, and used traces that had been pre-processed by the Bro Network Intrusion Detection System [13, 12].

We did not observe `FTP-DATA`, `finger`, and `IDENT` connections as these connections are the result of local hosts responding to remote hosts, and are not likely to be accepted by a host that has not issued a request for such a connection. These connections are thus unlikely to be useful for worm propagation.

5 Results

Table 2. Alarms reported by reverse sequential hypothesis testing combined with credit-based rate limiting. The cause of each alarm was later identified manually by comparing observed traffic to signature behaviors described at online virus libraries

	`isp-03`	`isp-04`
Worms/Scanners detected		
`CodeRed II`	2	0
`Blaster`	0	1
`MyDoom`	0	3
`Minmail.j`	0	1
HTTP (other)	3	1
Total	5	6
False alarms		
HTTP	0	3
SMTP	0	3
Total	0	6
P2P detected	6	11
Total identified	11	23

Table 3. Alarms reported by virus throttling

	`isp-03`	`isp-04`
Worms/Scanners detected		
`CodeRed II`	2	0
`MyDoom`	0	1
HTTP (other)	1	1
Total	3	2
False alarms	0	0
P2P detected	2	3
Total identified	5	5

Our reverse sequential hypothesis test detected two hosts infected with `CodeRed II` [4, 20] from the April, 2003 trace (`isp-03`). Our test detected one host infected

Table 4. Composite results for both traces. A total of 7 HTTP scanning worms and 5 email worms were present

	Alarms	Detection	Efficiency	Effectiveness
$\overline{HT}$	34	11	0.324	0.917
virus-throttling	10	5	0.500	0.417

Table 5. Comparison of rate limiting by credit-based connection rate limiting (CBCRL) vs. a virus throttle. Unnecessary rate limiting means that CBCRL dropped at least one packet from a host. For virus throttling, we only classify a host as rate limited if the delay queue reaches a length greater than five

	CBCRL		Virus Throttling	
	`isp-03`	`isp-04`	`isp-03`	`isp-04`
Worms/Scanners	5	1	3	4
P2P	4	8	3	7
Unnecessary rate limiting	0	0	84	59

with `Blaster/Lovsan` [5], three hosts infected with `MyDoom/Novarg` [11, 21], and one host infected with `Minmail.j` [6] from the January, 2004 trace (`isp-04`). The worms were conclusively identified by painstakingly comparing the logged traffic with the cited worm descriptions at various online virus/worm information libraries. Our test also identified four additional hosts that we classify as HTTP scanners because each sent SYN packets to port 80 of at least 290 addresses within a single class B network. These results are summarized in Table 2.

While peer-to-peer applications are not necessarily malicious, many network administrators would be loathe to classify them as benign. Peer-to-peer file sharing applications also exhibit ambiguous network behavior, as they attempt to contact a large number of transient peers that are often unwilling or unavailable to respond to connection requests. While peer-to-peer clients are deemed undesirable on most of the corporate networks that we envision our approach being used to protect, it would be unfair to classify these hosts as infected. For this reason we place hosts that we detect running peer-to-peer applications into their own category. Even if detections of these hosts are classified as false alarms, the number of alarms is manageable.

Three additional false alarms were reported for three of the 60 (`isp-04`) total hosts transmitting SMTP traffic. We suspect the false alarms are the result of bulk retransmission of those emails that have previously failed when the recipients' mail servers were unreachable. We suggest that organizations may want to white-list their SMTP servers, or significantly increase the detection thresholds for this protocol.

The remaining three false alarms are specific to the `isp-04` trace, and resulted from HTTP traffic. It appears that these false alarms were raised because of a temporary outage at a destination network at which multiple remote hosts became unresponsive. These may have included servers used to serve inlined images.

Table 6. The number of first-contact connections permitted before hosts were reported as infected. The value pairs represent individual results for two different `CodeRed II` infections and two different HTTP scanners

	$\overleftarrow{HT}$ with CBCRL	Virus Throttling
`CodeRed II`	10,10	6,7
Other HTTP scanners	10,10	102,526

Upon discovering these failures, we came to realize that it would be possible for an adversary to create Web sites that served pages with large numbers of inlined image tags linked to non-responsive addresses. If embedded with scripts, these sites might even be designed to perform scanning of the client's network from the server. Regardless, any client visiting such a site would appear to be engaged in HTTP scanning. To prevent such denial of service attacks from rendering a worm detection system unusable, we require a mechanism for enabling users to deactivate quarantines triggered by HTTP requests. We propose that HTTP requests from such hosts be redirected to a site that uses a CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart [23]), to confirm that a user is present and was using a Web browser at the time of quarantine.

Results for our implementation of Twycross and Williamson's virus throttle [22] are summarized in Table 3. Their algorithm blocked both instances of `CodeRed II`, but failed to detect `Blaster`, three instances of `MyDoom` (which is admittedly an email worm and not an IP scanning worm), and two low rate HTTP scanners. It did, however, detect one host infected with `MyDoom` that $\overleftarrow{HT}$ failed to detect. The virus throttle also detected fewer hosts running peer-to-peer applications, which for fairness we classify as a reduction in false alarms in virus throttling's favor in our composite results summarized in Table 4.

These composite results for both traces report the number of hosts that resulted in alarms and the number of those alarms that were detections of the 12 worms located in our traces. We also include the *efficiency*, which is the number of detections over the total number of alarms, and the *effectiveness*, which is the total number of detections over the total number of infected hosts we have found in these traces. While $\overleftarrow{HT}$ is somewhat less efficient than virus throttling, the more than two-fold increase in effectiveness is well worth the trade-off. In addition, corporate networks that forbid peer-to-peer file sharing applications will see a two-fold increase in efficiency.

Table 5 shows the number of hosts that had connection requests blocked by our credit-based algorithm and the number of hosts that were rate limited by Twycross and Williamson's algorithm. For credit-based connection rate limiting, we say that a machine has been rate limited if a single packet is dropped. For the virus throttle, we say that a machine has been rate limited if the outgoing delay queue length is greater than five, giving Twycross and Williamson the benefit of the doubt that users won't notice unless connections are severely throttled. Our credit-based algorithm only limited the rates of hosts that our

reverse sequential hypothesis test reported as infected. In contrast, even given our generous definition, more than 10% of the hosts in both traces were rate limited by Twycross and Williamson's algorithm.

Table 6 reports the number of first-contact connections permitted by the two approaches for those scanners that both detected. `CodeRed II` is a fast scanner, and so virus throttling excels in blocking it after 6 to 7 connection requests. This speed is expected to come at the price of detecting any service, malicious or benign, that issues high-rate first-contact connections.

Reverse Sequential Hypothesis Testing with credit-based connection rate limiting detects worms after a somewhat higher number of first-contact connections are permitted (10), but does so regardless of the scanning rate. Whereas our approach detects a slow HTTP scanner after 10 first-contact connection requests, the virus throttle requires as many as 526.

6 Limitations

Credit-based connection rate limiting is resilient to network uplink outages as hosts starved for credits will receive an allowance credit seconds after the network is repaired. Unfortunately, this will be of little consolation as Reverse Sequential Hypothesis Testing($\overleftarrow{HT}$) may have already concluded that all hosts are scanners. This may not be a problem if network administrators are given the power to invalidate observations made during the outage period, and to automatically reverse any quarantining decisions that would not have been taken without these invalid observations.

Of greater concern is that both Reverse Sequential Hypothesis Testing and credit-based connection rate limiting rely exclusively on the observation that hosts engaged in scanning will have lower first-contact connection success rates than benign hosts. New hypotheses and tests are required to detect worms for which this statistical relationship does not hold.

In particular, our approach is not likely to detect a *topological* worm, which scans for new victim hosts by generating a list of addresses that the infected host has already contacted. Nor is our approach likely to detect flash worms, which contain hit-lists of susceptible host addresses identified by earlier scans.

Also problematic is that two instances of a worm on different networks could collaborate to ensure that none of their first-contact connections will appear to fail. For example, if worm A does not receive a response to a first-contact connection request after half the timeout period, it could send a message to worm B asking it to forge a connection response. This *forged response attack* prevents our system from detecting connection failures. To thwart this attack for TCP connections, a worm detection system implemented on a router can modify the TCP sequence numbers of traffic as it enters and leaves the network. For example, the result of a hash function $h(\mathrm{IP}_{\mathrm{local}}, \mathrm{IP}_{\mathrm{remote}}, \mathrm{salt})$ may be added to all sequence numbers on outgoing traffic and subtracted from all incoming sequence numbers. The use of the secret salt prevents the infected hosts from calculating the sequence number used to respond to a connection request which

they have sent, but not received. By storing the correct sequence number in the `FCC` queue, responses can then be validated by the worm detection system.

Another concern is the possibility that a worm could arrive at its target already in possession of a list of known repliers – hosts that are known to reply to connection requests at a given port. This *known-replier attack* could employ lists that are programmed into the worm at creation, or accumulated by the worm as it spreads through the network. First-contact connections to these known-repliers will be very likely to succeed and can be interleaved with scans to raise the first-contact connection success rate. A one to one interleaving is likely to ensure that more than half of all connections succeed. This success rate would enable the scanner to bypass credit-based connection rate limiting, and delay detection by Reverse Sequential Hypothesis Testing until the scanner had contacted all of its known-repliers. What's worse, a worm could avoid detection altogether if the detection system defines a first-contact connection with respect to a fixed sized previously contact host (`PCH`) set. If the `PCH` set tracks only the n previously visited hosts, the scanner can cycle through $(n/2)+1$ known-repliers, interleaved with as many new addresses, and *never* be detected[6]. To prevent a worm from scanning your local network by interleaving connections to known-repliers outside of your network, Weaver *et al.* [26] propose that one hypothesis test be run for local connections (i.e. those within the same IP block) and another for connections to remote hosts. If hosts in your local network are widely and randomly dispersed through a large IP space[7], then a worm will have a low probability of finding another host to infect before being quarantined.

A worm might also avoid detection by interleaving scanning with other apparently benign behavior, such as Web crawling. A subset of these *benign interleaving attacks* can be prevented by detecting scanners based on the destination port they target in addition to the source IP of the local host. While it is still fairly easy to create benign looking traffic for ports such as HTTP, for which one connection can lead to information about other active hosts receptive to new connections, this is not the case for ports such as those used by SSH. Running separate scan detection tests for each destination port that a local host addresses can ensure that connections to one service aren't used to mask scans to other services.

Finally, if an infected host can impersonate other hosts, the host could escape quarantine and cause other (benign) hosts to be quarantined. To address these *address impersonation attacks*, it is important that a complete system for network quarantining include strong methods for preventing IP masquerading by its local hosts, such as switch level egress filtering. Host quarantining should also be enforced as close to the host as is possible without relying on the host to quarantine itself. If these boundaries cannot be enforced between each host,

[6] For detecting such a worm, a random replacement policy will be superior to an LRU replacement policy, but will still not be effective enough for long known-replier lists.

[7] Randomly dispersing local hosts through a large IP space can be achieved by using a network address translation (NAT) switch.

one must assume that when one machine is infected, all of the machines within the same boundary will also be infected.

7 Related Work

We were motivated by the work of Moore *et al.* [10], who model attempts at containing worms using quarantining. They perform theoretical simulations, many of which use parameters principally from the `CodeRed II` [4, 20] outbreak. They argue that it is impossible to prevent systems from being vulnerable to worms and that treatment cannot be performed fast enough to prevent worms from spreading, leaving containment (quarantining) as the most viable way to prevent worm outbreaks from becoming epidemics.

Early work on containment includes Staniford *et al.*'s work on the GrIDS Intrusion Detection System [19], which advocates the detection of worms and viruses by tracing their paths through the departments of an organization. More recently, Staniford [16] has worked to generalize these concepts by extending models for the spread of infinite-speed, random scanning worms through homogenous networks divided up into 'cells'. Simulating networks with 2^{17} hosts (two class B networks), Staniford limits the number of first-contact connections that a local host initiates to a given destination port to a threshold, T. While he claims that for most ports, a threshold of $T = 10$ is achievable in practice, HTTP and KaZaA are exceptions. In comparison, reverse sequential hypothesis testing reliably identifies HTTP scanning in as few as 10 observations.

The TRAFEN [2, 3] system also observed failed connections for the purpose of identifying worms. The system was able to observe larger networks, without access to end-points, by inferring connection failures from ICMP messages. One problem with acting on information at this level is that an attacker could spoof source IP addresses to cause other hosts to be quarantined.

Our use of rate limiting in order to buy time to observe worm behavior was inspired by the virus throttle presented by Twycross and Williamson [22], which we described in detail in Section 3. Worms can evade a throttle by scanning at rates below one connection per second, allowing epidemics to double in size as quickly as once every two seconds.

An approach quite similar to our own has been simultaneously developed by Weaver, Staniford, and Paxson [26]. Their approach combines the rate limiting and hypothesis testing steps by using a reverse sequential hypothesis test that (like our CBCRL algorithm) assumes that connections fail until they are proven to succeed. As with CBCRL, out-of-order processing could cause a slight increase in detection delay, as the successes of connections sent before an infection event may be processed after connections are initiated after the infection event. In the context of their work, in which the high-performance required to monitor large networks is a key goal, the performance benefits are likely to outweigh the slight cost in detection speed.

For a history and recent trends in worm evolution, we recommend the work of Kienzle and Elder [8]. For a taxonomy of worms and a review of worm terminology, see Weaver *et al.* [25].

8 Future Work

As worm authors become aware of the limitations discussed in Section 6, it will be necessary to revise our algorithms to detect scanning at the resolution of the local host (source address) and targeted service (destination port), rather than looking at the source host alone. Solutions for managing the added memory requirements imposed by this approach have been explored by Weaver, Staniford, and Paxson [26].

The intrusiveness of credit-based connection rate limiting, which currently drops outgoing connection requests when credit balances reach zero, can be further reduced. Instead of halting outgoing TCP first-contact connection requests from hosts that do not maintain a positive credit balance, the requests can be sent immediately and the responses held until a positive credit balance is achieved. This improvement has the combined benefits of reducing the delays caused by false rate limiting while simultaneously ensuring that fewer connections are allowed to complete when a high-speed scanning worm issues a burst of connection requests. As a result, the remaining gap in response speed between credit-based connection rate limiting and Twycross and Williamson's virus throttle can be closed while further decreasing the risk of acting on false positives.

Finally, we would like to employ additional indicators of infection to further reduce the number of first-contact connection observations required to detect a worm. For example, it is reasonable to conclude that, when a host is deemed to be infected, those hosts to which it has most recently initiated successful connections are themselves more likely to be infected (as was the premise behind GrIDS [19]). We propose that this be accomplished by adding an event type, the report of an infection of a host that has recently contacted the current host, to our existing hypothesis test.

9 Conclusion

When combined, credit-based connection rate limiting and reverse sequential hypothesis testing ensure that worms are quickly identified with an attractively low false alarm rate. While no system can detect all possible worms, our new approach is a significant improvement over prior methods, which detect a smaller range of scanners and unnecessarily delay network traffic. What's more, the techniques introduced in this paper lend themselves to efficient implementation, as they need only be activated to observe a small subset of network events and require little calculation for the common case that traffic is benign.

10 Acknowledgments

This paper could not have been completed without the continued support of Vern Paxson and Hari Balakrishnan. We are indebted to Dave Andersen and Noah Case for the network logs used for our analysis. We would also like to thank the anonymous reviewers as well as Nick Feamster, David Molnar, Rodrigo Miragaia Rodrigues, David Savitt, Matt Williamson, and especially Glenn Holloway for taking the time to review and comment on earlier drafts of this paper. Stuart Schechter would like to thank the National Science Foundation for support under grant CCR-0310877.

References

1. George Bakos and Vincent Berk. Early detection of internet worm activity by metering ICMP destination unreachable messages. In *Proceedings of the SPIE Aerosense*, 2002.
2. Vincent Berk, George Bakos, and Robert Morris. Designing a framework for active worm detection on global networks. In *Proceedings of the IEEE International Workshop on Information Assurance*, March 2003.
3. Vincent H. Berk, Robert S. Gray, and George Bakos. Using sensor networks and data fusion for early detection of active worms. In *Proceedings of the SPIE Aerosense Conference*, April 2003.
4. CERT. "Code Red II:" another worm exploiting buffer overflow in IIS indexing service DLL. http://tinyurl.com/2lzgb.
5. F-Secure. Computer virus information pages: Lovsan. http://tinyurl.com/jozm.
6. F-Secure. Computer virus information pages: Mimail.j. http://tinyurl.com/3ybsp.
7. Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan. Fast portscan detection using sequential hypothesis testing. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 9–12, 2004.
8. Darrell M. Kienzle and Matthew C. Elder. Recent worms: a survey and trends. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 1–10. ACM Press, October 27, 2003.
9. David Moore, Vern Paxson, Stefan Savage, Colleen Shannon, Stuart Staniford, and Nicholas Weaver. Inside the Slammer worm. *IEEE Security and Privacy*, 1:33–39, July 2003.
10. David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *Proceedings of IEEE INFOCOM*, April 1–3 2003.
11. Network Associates Inc. Security threat report for W32/MydoomMM. http://tinyurl.com/2asgc.
12. Vern Paxson. Bro: A system for detecting network intruders in real-time. http://www.icir.org/vern/bro-info.html.
13. Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23–24):2435–2463, 1999.
14. Stelios Sidiroglou and Angelos D. Keromytis. Countering network worms through automatic patch generation. Technical Report CUCS-029-03, 2003.
15. Stelios Sidiroglou and Angelos D. Keromytis. A network worm vaccine architecture. In *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), Workshop on Enterprise Security*, June 2003.

16. Stuart Staniford. Containment of scanning worms in enterprise networks. *Journal of Computer Security*, Forthcoming.
17. Stuart Staniford, James Hoagland, and Joseph McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1):105–136, 2002.
18. Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, August 7–9, 2002.
19. S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle. GrIDS – A graph-based intrusion detection system for large networks. In *Proceedings of the 19th National Information Systems Security Conference*, volume 1, pages 361–370, October 1996.
20. Symantec. Security response – CodeRed II. http://tinyurl.com/89t0.
21. Symantec. Security response – W32.Novarg.A@mm. http://tinyurl.com/2qbaj.
22. Jamie Twycross and Matthew M. Williamson. Implementing and testing a virus throttle. In *Proceedings of the 12th USENIX Security Symposium*, August 4–8 2003.
23. Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart (automatically) or how lazy cryptographers do AI. Technical Report CMU-CS-02-117, February 2002.
24. Abraham Wald. *Sequential Analysis*. J. Wiley & Sons, New York, 1947.
25. Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 11–18. ACM Press, October 27, 2003.
26. Nicholas Weaver, Stuart Staniford, and Vern Paxson. Very fast containment of scanning worms. In *Proceedings of the 13th USENIX Security Symposium*, August 9–13 2004.
27. Matthew M. Williamson. Throttling viruses: Restricting propagation to defeat malicious mobile code. In *Proceedings of The 18th Annual Computer Security Applications Conference (ACSAC 2002)*, December 9–13, 2002.

A Optimizing the Computation of Repeated Reverse Sequential Hypothesis Tests

It is unnecessarily expensive to repeatedly recompute Λ in reverse sequence each time a new first-contact connection is observed. A significant optimization requires that we maintain single state variable $\bar{\Lambda}$, calculated iteratively in the order in which events are observed.

$$\bar{\Lambda}(\mathbf{Y}_n) \;=\; \max\big(1,\; \bar{\Lambda}(\mathbf{Y}_{n-1})\phi(Y_n)\big) \quad \bar{\Lambda}(\mathbf{Y}_0) \;\equiv\; 1. \tag{1}$$

We will prove that $\bar{\Lambda}(\mathbf{Y}_n) > \eta_1$ if and only if a reverse sequential hypothesis test starting backward from observation n would lead to infection conclusion.

We first prove the following lemma stating that if a reverse sequential hypothesis test reports an infection, our optimized algorithm will also report an infection.

Lemma 1. *For $\eta_1 > 1$ and for mutually independent random variables Y_i,*

$$\forall m \in [1,n] : \Lambda(Y_n, Y_{n-1}, \ldots, Y_m) \geq \eta_1 \Rightarrow \bar{\Lambda}(\boldsymbol{Y}_n) \geq \eta_1 \tag{2}$$

Proof. We begin by replacing the Λ term with its equivalent expression in terms of ϕ:

$$\eta_1 \leq \Lambda(Y_n, Y_{n-1}, \ldots, Y_m) \tag{3}$$

$$\leq \prod_{i=m}^{n} \phi(Y_i) \tag{4}$$

We can place a lower bound on the value of $\bar{\Lambda}(Y_n)$ by exploiting the fact that, in any iteration, $\bar{\Lambda}$ cannot return a value less than 1.

$$\begin{aligned}\bar{\Lambda}(\mathbf{Y}_n) &= \bar{\Lambda}(Y_1, Y_2, \ldots, Y_n) \\ &\geq 1 \cdot \bar{\Lambda}(Y_m, Y_{m+1}, \ldots, Y_n) \\ &\geq \prod_{i=m}^{n} \phi(Y_i) \geq \eta_1\end{aligned}$$

where the last inequality follows the steps taken in Equations (3) and (4).

Thus, $\Lambda(Y_n, Y_{n-1}, \ldots, Y_m) \geq \eta_1 \Rightarrow \bar{\Lambda}(\mathbf{Y}_n) \geq \eta_1$.

We must also prove that our optimized algorithm will only report an infection when a reverse sequential hypothesis test would also report an infection. Recall that a reverse sequential hypothesis test will only report an infection if Λ exceeds η_1 before falling below η_0.

Lemma 2. *For thresholds $\eta_0 < 1 < \eta_1$ and for mutually independent random variables Y_i, if $\bar{\Lambda}(\boldsymbol{Y}_i) \geq \eta_1$ for some $i = n$, but $\bar{\Lambda}(\boldsymbol{Y}_i) < \eta_1$ for all $i \in [1, n-1]$, then there exists a subsequence of observations starting at observation n and moving backward to observation $m \in [1,n]$ for which $\Lambda(Y_n, Y_{n-1}, \ldots, Y_m) \geq \eta_1$ and such that there exists no k in [m,n] such that $\Lambda(Y_n, Y_{n-1}, \ldots, Y_k) \leq \eta_0$*

Proof. Choose m as the largest observation index for which it held that:

$$\bar{\Lambda}(\mathbf{Y}_{m-2})\phi(Y_{m-1}) < 1$$

We know that $m < n$ because $\bar{\Lambda}(\mathbf{Y}_{n-1})\phi(Y_n)$ is greater than η_1 which is in turn greater than 1. Let $m = 1$ if the above relation does not hold for any observation with index greater than 1. It follows that $\bar{\Lambda}(\mathbf{Y}_{m-1}) = 1$ and thus:

$$\bar{\Lambda}(\mathbf{Y}_m) = \phi(Y_m)$$

Because we chose m such that $\bar{\Lambda}(\mathbf{Y}_{j-2})\phi(Y_{j-1}) \geq 1$ for all $j > m$:

$$\begin{aligned}\bar{\Lambda}(\mathbf{Y}_n) &= \prod_{j=m}^{n} \phi(Y_j) \\ &= \Lambda(Y_n, Y_{n-1}, \ldots, Y_m)\end{aligned}$$

Thus, $\bar{\Lambda}(\mathbf{Y}_n) \geq \eta_1 \Rightarrow \Lambda(Y_n, Y_{n-1}, \ldots, Y_m) \geq \eta_1$.

To prove that there exists no k in $[m, n]$ such that $\Lambda(Y_n, Y_{n-1}, \ldots, Y_k) \leq \eta_0$, suppose that such a k exists. It follows that:

$$\prod_{j=k}^{n} \phi(Y_j) \leq \eta_0 < 1 \tag{5}$$

Recall that we chose m to ensure that:

$$\eta_1 \leq \prod_{j=m}^{n} \phi(Y_j) \tag{6}$$

The product on the right hand side can be separated into factors from before and after observation k.

$$\eta_1 \leq \prod_{j=m}^{k-1} \phi(Y_j) \cdot \prod_{j=k}^{n} \phi(Y_j) \cdot \tag{7}$$

We then use Equation (5) to substitute an upper bound of 1 on the latter product.

$$\begin{aligned}\eta_1 &\leq \prod_{j=m}^{k-1} \phi(Y_j) \\ \eta_1 &\leq \bar{\Lambda}(\mathbf{Y}_{k-1})\end{aligned}$$

This contradicts the hypothesis that $\bar{\Lambda}(\mathbf{Y}_i) < \eta_1$ for all $i \in [1, n-1]$.

If we were concerned with being notified when the test came to the 'benign' conclusion we could create an analogous function $\underline{\Lambda}$:

$$\underline{\Lambda}(\mathbf{Y}_n) \;=\; \min\left(1,\; \underline{\Lambda}(\mathbf{Y}_{n-1})\phi(Y_n)\right)$$

The lemmas required to show equivalence and proof are also analogous.