Scanner Detection Based on Connection Attempt Success Ratio with Guaranteed False Positive and False Negative Probabilities

Seung Yeob Nam and Hyong S. Kim

June 20, 2006
CMU-CyLab-06-011

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Scanner Detection Based on Connection Attempt Success Ratio with Guaranteed False Positive and False Negative Probabilities

Seung Yeob Nam and Hyong S. Kim
Dept. of ECE and CyLab, Carnegie Mellon University
Pittsburgh, PA 15213
synam@andrew.cmu.edu, kim@ece.cmu.edu

***Abstract*— Since the link rate is very high up to 40Gbps these days, scanning packets can spread very fast. At this high speed, only a small chance of missing on-going scanning activity can lead to catastrophic results. Thus, fast and accurate detection of scanners is a very important problem. High-speed packet processing usually requires high-speed memory, SRAM, and the size of SRAM is very limited compared with DRAM. We propose a connection attempt success ratio based scanning detection scheme which guarantees false positive and false negative probabilities under a memory-limited environment. Our scheme can also detect slow scanners with guaranteed performance. A sampling-based extended version can overcome the limitation of short-history-based scanning detection schemes and detects enhanced scanners with a list of pre-acquired IP addresses with guaranteed performance. The proposed scheme reduces the required memory size from O(N^2) to O(N), where N is the number of active hosts. We apply Bloom filter in order to further reduce the memory size. We evaluate the performance of the proposed scheme through simulation.**

Keywords – *scanner, slow scanner, scanner detection, connection attempt success ratio, Bloom filter*

I. INTRODUCTION

Internet attacks such as distributed denial-of-service (DDoS) attacks and worm attacks are increasing in severity. According to CERT [1], the number of reported incidents by network attack has grown almost exponentially and it continues to increase. Computer worms and bots are substantial threats to large networks because they can spread very rapidly and used for DDoS [2]. The first phase of worms and bots begin with scanning of vulnerable hosts. In this paper, we address the detection of such scanners in order to prevent further attacks and damage.

There are many important issues that need to be considered in order to make a practical scanning detection system. First, memory constraints should be considered. Since inter-packet arrival time can be very short due to a high link rate, SRAM is usually required for fast packet processing [3]. Thus, the required memory size needs to be small enough to fit in SRAM in this case. Second, low false positive and false negative probabilities need to be guaranteed since at a very high link speed only a small chance of missing on-going scanning activity can lead to catastrophic damages. Third, the detection algorithm should be simple and the number of memory accesses should be small so that it can cope with bursts of packets arriving at a high speed. Fourth, fast detection is important because the damage can be minimized with early detection. Fifth, detection of slow scanners is also important since most of window-based schemes can be defeated by slow scanners. There have been many approaches to the scanner detection problem [3-12], but there is no scheme addressing all the above issues collectively.

We can classify most of existing schemes into four categories depending on the metric used to detect scanners. The metrics are connection attempts [7], connection attempt rate [4, 5, 6], connection attempt failure rate [8, 11], connection attempt failure ratio [3, 9, 10]. Among them, connection attempt-based schemes, e.g. Bro [7], can detect slow scanners, but they usually have high false positive rates. In case of either connection attempt rate-based scheme or connection failure rate-based scheme, the scanners can evade detection by lowering the scanning rate if the threshold is estimated by them. A low detection threshold may incur many false alarms, and thus, determining the threshold is very difficult problem. Thus, we use the statistic of connection attempt success ratio to detect scanners. But, our statistic is different from the ones used in [9, 10] since we directly use the statistics of

connection attempt success ratio, while Jung *et al.* [9] and Schechter *et al*. [10] use a likelihood ratio for sequential hypothesis testing or reverse sequential hypothesis testing.

Based on the connection attempt success ratio, our scheme addresses five important issues stated above. In more detail, the contribution of this paper can be summarized as follows:

- Threshold Random Walk (TRW) [9] and the optimized TRW [10] are the state-of-the-art technique in the scanning detection area and they can guarantee the false positive and false negative probabilities when there is no memory constraint. Since TRW needs to parse every arriving packet, SRAM would be required for high speed packet processing [3]. Since the size of SRAM is highly limited to around tens or hundreds of Megabytes, any scheme using SRAM is subjected to memory conflict issues such as collisions in hash tables. The original TRW scheme [9] and the optimized TRW [10] are not addressing this memory issues and these schemes may not guarantee the false positive and false negative probabilities under the memory-limited environment. Weaver *et al*. [3] propose a modified version of the TRW scheme considering this memory issues, but this modified scheme does not guarantee the false positive and false negative probabilities [3]. The proposed scheme is designed considering this memory conflict issues, and thus, it can guarantee the false positive and false negative probabilities under a memory-limited environment. This means that the proposed scheme is the first approach which attempts to guarantee the required performance under the realistic memory-limited environment.
- It is said that per-flow state needs to be kept in order to detect stealthy port-scanning [11]. When N is the total number of active hosts, the storage space of $O(N^2)$ will be required in order to detect slow scanners. However, in our scheme, a decision is made based on a rather small, less than 25, and fixed number of connection attempts. Thus, each source address occupies only a fixed size of memory and the memory size is kept on the order of $O(N)$.
- Since our scheme is not time window-based and the connection log of a source IP with a low connection success ratio can be kept longer than those of other source addresses with higher connection success ratios, our scheme can detect slow scanners. Although there are some other schemes which can detect slow or stealthy scanners [7, 12], the proposed scheme is the first one which detects slow scanners with guaranteed performance under a memory-limited environment.
- Many scanning detection schemes usually assume malicious hosts are doing random scanning without any knowledge of valid IP addresses. But, our sampling-based extended scheme can detect enhanced scanners which has a *limited* number of valid IP addresses in advance. However, if the malicious hosts have the complete lists of assigned IP addresses like complete scan in [13] or Flash Worm in [14], then it would be very difficult to detect such scanners based on their connection activities. Detection of such scanners is out of the scope of this paper.

Only assumption for the guarantee of false positive and false negative probabilities is that the malicious scanner and benign scanner have distinct behaviors in terms of the connection success ratio. We apply *Bloom filter* in order to reduce the size of memory required for connection state information. We find that 16 MB SRAM is adequate to handle millions of flows in core routers [15].

The rest of the paper is organized as follows. In Section II, we discuss related work. In Section III, we explain the proposed scanner detection scheme and the structure of the hash table used to store connection information in detail. In Section IV, we analytically derive the detection threshold which can guarantee the performance of our scheme in terms of false positive and false negative probabilities under collisions in Bloom filters. In Section V, we extend our detection scheme by incorporating a sampling technique to complement the drawback of the basic version. In Section VI we evaluate the performance of the proposed scheme through simulation. Finally, conclusions are given in Section VII.

II. RELATED WORK

Historically most scan detection schemes detect N events within a time interval of T seconds. The Network Security Monitor (NSM) [5] detect any source IP address connecting to more than 15 distinct destinations within a given time window. Snort [6] also implements similar methods. If the detection threshold or the measurement time interval is known to the malicious hosts, then the attackers can evade the detection by simply decreasing the

scanning rate. Bro [7] manages the number of distinct destination addresses for each source address and alerts an alarm if the number reaches a threshold. Without the time window, the Bro could detect slow scanners at the risk of non-negligible false alarms and it would require very large memory.

Large number of false positive occurs since there are many legitimate scanners such as in Internet search engine applications. The connection failures are known to be a better metric for detection of scanners. Robertson *et al.* focus on failed connection attempts, using a threshold method [8]. Determination of a good threshold is a challenging problem. Low threshold generates excessive false positives. High threshold may miss many slow scanners. Robertson *et al.* show that the performance varies greatly based on the threshold value.

Spice [12] manages the probability distribution of normal traffic and detects scans, including stealthy scans, based on the deviations from the normal behavior. But, it may not be suitable for fast detection on high-speed links since it requires too much computation and selecting the threshold for the anomalous event reporting is very difficult since traffic characteristics varies from site to site and over time, too.

Jung *et al.* propose Threshold Random Walk (TRW) detection scheme [9] and Schechter *et al*. propose optimized TRW scheme [10]. Both of them detect the scanning activity rapidly with a small number of connection attempts, usually less than 20, with guaranteed performance. However, memory constraint is not considered in both cases. Thus, false positive and false negative probabilities may not be guaranteed under a memory-limited environment, e.g. collisions in the connection status table or limited space for each source IP address. Weaver et al. [3] uses a simplified version of TRW in order to reduce the complexity. However, it does not guarantee the false positive and negative rates due to collisions in the connection cache [3].

There are also some approaches focusing on network attacks with reduced memory size [4, 11, 16]. Kompella *et al.* [11] attempts to detect scanning as well as some other network attacks using partial completion filters without keeping per-flow states. The connection failure rate is calculated from the difference the number of TCP SYN packets and TCP FIN packets. Determining the detection threshold is usually a difficult problem and it is not easy to detect slow scanners because the contribution of slow scanners to difference metric is not likely to be significant.

Venkataraman *et al*. propose a scheme to detect superspreaders using connection attempt rates or connection failure rates [4]. It uses a sampling technique in order to reduce the packet processing and the memory size. Sampling could reduce the number of packets to be processed by a few orders of magnitude and leads to simpler implementation. Detection of scanners based on connection attempt or connection failure rates could be evaded by slow scanners with a low probing rate. We use this sampling technique in the extended version in order to reduce the load for the connection table and overcome the limitation of the short-history based scanner detection scheme.

Estan *et al.* propose a family of efficient bitmap algorithm for counting active flows [16]. Triggered bitmap may be useful to count the number of connection attempts for each source address with a moderate error. But, in our scheme we make a decision with a rather small number of connection attempts and responses, less than 25. Since the number is usually small, the triggered bitmap is likely to be reduced to the direct bitmap. The direct bitmap can be considered as a special case of Bloom filters [17, 18] with only one hash function. In our scheme, accuracy in counting is very important to guarantee false positive and false negative probabilities and the accuracy can usually be improved with more than one hash functions under the same memory size. Thus, we use Bloom filters to count connection attempts and responses.

III. SCANNER DETECTION SCHEME

We now describe the operation of the proposed scanner detection scheme in detail.

A. *Detection Rule*

The scanner is detected based on the connection attempt success ratio of the source host. The connection attempt success ratio of a source s, $q(s)$, is defined as follows.

$$q(s) = \frac{RESPONSE(s)}{ATTEMPT(s)}, \tag{1}$$

where the *ATTEMPT(s)* is the number of distinct IP addresses that a source s attempts to connect to and the *RESPONSE(s)* is the number of distinct IP addresses that responded to the source s. We define two timers,

TIMER1(s) and *TIMER2(s)*. The scanner detection decision is made after observing n connection attempts for each source address. For the n-th connection attempt, our scheme waits up to *TIMER1(s)* for its response. *TIMER1(s)* is used to allow sufficient time for the response to the last attempt to be counted. According to [19], the median round-trip time (rtt) is measured to be lower than 450 *msec* by IPMON system of Sprint even including transcontinental connections. We set the value of *TIMER1(s)* to 500 *msec*. *TIMER2(s)* tracks how long s has been idle since its last connection attempt. *TIMER2(s)* is used to delete source addresses which have a high connection success ratio and have been idle for a long period when the memory storing connection status information is overloaded because these addresses are likely to be innocent. When the *ATTEMPT(s)* reaches a pre-selected threshold n and the *TIMER1(s)* expires, then we detect the source as a scanner if the following condition is satisfied:

$$q(s) \le \eta , \tag{2}$$

where η is the detection threshold. Variable parameters n and η are determined according to the required false positive and false negative probabilities and it will be discussed in Sections IV and V.

B. Implementation

We address the implementation complexity and the memory and processing requirement for the proposed scheme. We first describe the data structure of the scheme and then the connection status update procedure.

1) Bloom filters

Using the statistic of connection failure ratio, the proposed scheme has to manage the connection status for each source and destination IP address pair[1]. Although we allocate a connection status-related hash table with a million entries, occasionally the total number of connections including connection attempts can exceed the limit. Weaver et al. [3] considers the use of a million-entry connection cache which tracks the connection status in each direction between a pair of hosts. When the total number of connection attempts exceeds a million or different host pairs hash to the same entry of the connection cache, two different connection states are merged into a single entry. This aliasing policy increases false negatives as failed connection attempts of different pairs are used jointly. According to [3], if the table is 20% full, then they fail to detect roughly 20% of individual scanning attempts. The false negatives may not be guaranteed due to such failures. In order to avoid such limitation, we manage connection states and data in the follow way using the Bloom filters.

If we allocate w bits for each source and destination address pair (s, d) in order to track its connection status, we then need the storage space of the order of $O(N^2)$ in the worst case, where N is the total number of valid IP addresses. However, if a fixed number of bits are allocated for each source address, then the memory size decreases to the order of $O(N)$. More specifically we reduce the memory space by storing a limited set of the distinct destination addresses for a source s in an m-bit Bloom filter [17, 18]. We define a vector $\boldsymbol{V}$ of m bits to contain the destination address information. Each element of the vector is initially set to 0. k independent hash functions, $h_1, h_2, \ldots, h_k$, each of which has a range of $\{1, \ldots, m\}$, are used to map each destination address d into the vector $\boldsymbol{V}$.

If a packet from the source s to the destination d is observed, then the bit vector corresponding to the source address s is determined by hashing the source address s. The destination address d is then registered in the bit vector as follows. The bits at positions $h_1(d), h_2(d), \ldots, h_k(d)$ in $\boldsymbol{V}$ are set to 1. Destination d of the first packet from s to d gets registered using the Bloom filter and the subsequent packets with the same (s, d) do not get registered. The bits at positions $h_1(d), h_2(d), \ldots, h_k(d)$ represent the existence of the prior packets of (s, d). However, it is possible that $h_1(d), h_2(d), \ldots, h_k(d)$ bits are set by other destinations as the Bloom filter gets filled. We define this blocking in the Bloom filter as a *collision*. One of the advantages of Bloom filters is that it is possible to control the probability of collision by adjusting parameters k and m.

If the number of destinations associated with the source address s is increased, then the number of bits m should be increased to keep the collision probability at the same level. In the proposed scheme, source s is determined to be a scanner or not if the number of distinct destinations reaches a pre-determined threshold n. Thus, the number of destination IP addresses associated with one source address does not exceed n and the bitmap size

[1] Although we consider only horizontal scanning over distinct IP addresses, our scheme can be readily extended to detection of vertical scanning over the distinct port numbers by considering the pair of destination address/port number instead of just destination address.

m can be kept constant. We analyze the effect of collisions on the false positive and false negative probabilities in Sections IV and V.

2) Data Structure

We manage the connection status, especially connection attempt status and response status, of each source address in a table called *connection status table*. Fig. 1 shows the detailed structure of the connection status table. The table can be considered as a hash table which is indexed by the hash value of the source IP address. If we let h_s denote this hash function, then a source address s can occupy an entry at the address of $h_s(s)$. Occasionally two or more different source addresses may have the same hash value. In order to cope with this collision, we put four entries for each hash value[2] and we put *IP address* (source address) field in each entry to discriminate different source addresses which have the same hash value. Thus, the position of the entry corresponding to a source address s is determined by the $(h_s(s), s)$ pair. Two bit vectors are allocated for each entry to manage connection attempt status and response status of the corresponding source address. For an entry corresponding to a source address s, $\boldsymbol{V_1}(s)$ is a bit vector which manages connection attempt status of s and $\boldsymbol{V_2}(s)$ is a bit vector which manages the status of responses to s. If the first packet from s to d is observed, then the address d is registered in the bit vector field $\boldsymbol{V_1}(s)$. If the response packet from d to s arrives following the request packet from s to d, then the address d is recorded in the bit vector field $\boldsymbol{V_2}(s)$. $\boldsymbol{V_1}$(s) manages the set of distinct destinations that the host s attempts to connect to and $\boldsymbol{V_2}(s)$ manages the set of destinations which responded to s.

As explained in the previous subsection, *ATTEMPT(s)* counts the number of attempts made by s and it is equal to the number of destinations registered in $\boldsymbol{V_1}(s)$. *RESPONSE(s)* counts the number of responses to s and it is equal to the number of addresses registered in $\boldsymbol{V_2}(s)$.

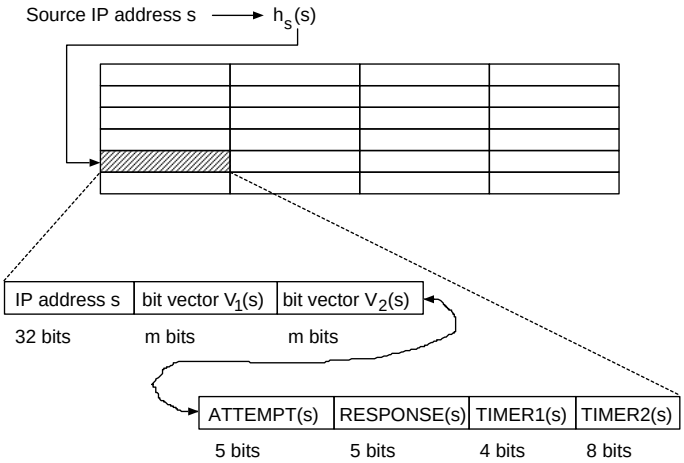

Fig. 1. The structure of connection status table

In case of TCP connection request through TCP SYN packet, no response will be sent back to the source if the destination IP address is inactive. If the SYN packet is sent to inactive port of an active host, then the host is likely to reply with TCP RST packet. If scanners usually send packets to invalid IP addresses, then we do not count TCP RST packets as a connection attempt failure. However, in the case of vertical scanning, we interpret TCP RST packets as a connection attempt failure. In the current scheme, only unanswered connection attempts are considered as the connection attempt failure. Our scheme can be extended so that TCP RST responses are counted as the connection attempt failure in such a scenario.

Two timers, *TIMER1(s)* and *TIMER2(s)*, are associated with the source s as discussed previously. If the *ATTEMPT(s)* reaches n, then *TIMER1(s)* is increased by 1 every 50 *msec*. This timer need not be increased individually for each source address s. All *TIMER1's* with the *ATTEMPT* value of n can be increased simultaneously or asynchronously within 50 *msec*. If we let t_1 denote the threshold for the *TIMER1(s)*, then the

[2] This structure can be implemented by either a linked list in software or a 4-way cache in hardware.

TIMER1(s) value of t_1 means that at least 50x(t_1-1) *msec* has passed since the arrival time of the last *n*-th attempt. Since we wait at least 500 *msec* after the arrival of the *n*-th attempt as discussed in the previous subsection, the default value of t_1 is set to 11. If the value of *TIMER1(s)* reaches t_1, then the decision on the scanning activity of *s* is made according to the rule described in the previous subsection. If the source address *s* is detected as a scanner, we report *s* as a scanner and clear every field corresponding to *s*. Otherwise, we just delete the entry of *s*.

TIMER2(s) is used to delete the entries of the seemingly innocent and less active, in terms of scanning activity, hosts when the connection status table is overloaded. The timer is reset to 0 whenever the value of *ATTEMPT(s)* is increased and it is increased by 1 every minute. The increment period of 1 minute is determined with the following reasoning. Let N_s denote the maximum number of source addresses which can be accommodated by the connection status table ($N_s = 10^6$ in the current setting). There are 4 entries for each address of the hash table. If we assume the hash function h_s is perfectly random, then a source address in one entry of the hash table sees the next different source address which maps to the same hash table address after about $N_s/4$ more distinct source addresses by the expectation of the geometric distribution. Fraleigh *et al*. [19] report that the average number of active flows per minute is less than 300,000 for all measured OC-48 links. The flow arrival rate is usually not higher than this rate and the arrival rate of distinct source addresses is uaually lower than the flow arrival rate. If the distinct-source-address arrival rate is as high as 250,000 per minute, then the interarrival time of different source addresses to the same hash address is about $N_s/4/250{,}000 = 1$ minute for $N_s = 10^6$. In order to discriminate the age of different source addresses mapping to the same hash address, the increment interval of *TIMER2(s)* needs to be not longer than 1 minute. Currently we select 1 minute. We need not put any upper limit on TIMER2(s), but 8 bits are allocated to the *TIMER2(s)* considering memory constraints. With 8 bits we can count up to 255 minutes. Since $N_s/4$ is usually a large number, it is not likely that more than 2 scanners map to the same hash address. In addition, innocent sources are not likely to stay long in the hash table due to the entry eviction policy explained in the next subsection. We expect that 8 bits are enough for *TIMER2(s)*.

3) Connection Status Table Update Procedure

The connection status table is updated when a packet arrives. Let us assume that a packet from *s* to *d* arrives at the monitoring node. Then, the connection status table is updated as follows.

Step 1: On the packet arrival, we check whether the packet is a response to an existing connection attempt. We check whether the address *s* is registered in both $\boldsymbol{V_1}(d)$ and $\boldsymbol{V_2}(d)$. If *s* exists in both $\boldsymbol{V_1}(d)$ and $\boldsymbol{V_2}(d)$, then the packet is a duplicate response to *d*. We just neglect the packet and exit the routine. If *s* exists in $\boldsymbol{V_1}(d)$, but not in $\boldsymbol{V_2}(d)$, then the packet is a new response for the attempt from *d*. We increase *RESPONSE(d)* by 1 and we register *s* in $\boldsymbol{V_2}(d)$. If *s* does not exist in either $\boldsymbol{V_1}(d)$ or $\boldsymbol{V_2}(d)$, then the packet is not a response and we go to Step 2.

Step 2: We check whether the arriving packet is a new connection attempt. If it's true, we register d in $\boldsymbol{V_1}(s)$. We check whether the destination *d* is registered in $\boldsymbol{V_1}(s)$. If *d* exists in $\boldsymbol{V_1}(s)$, then the packet is not a new connection attempt and we exit the routine. If *d* does not exist in $\boldsymbol{V_1}(s)$, the packet is a new attempt of *s*. Before registering *d* in $\boldsymbol{V_1}(s)$, we check whether the value of *ATTEMPT(s)* is *n* or not. If the threshold is reached, then we neglect the new attempt until the decision is made with the current data sets. If the threshold is not reached, we register *d* in $\boldsymbol{V_1}(s)$, increase the value of *ATTEMPT(s)* by 1, and exit the routine.

In the update procedure, there are 2 memory reads (one for the source address *s* and another for the destination address *d*) and 1 memory write at most. The processing overhead is thus adequate for high speed links.

In order to accommodate the hash collision, especially when more than 4 source addresses are mapped to the same hash address, we have the following eviction policy. We first evict the source address with the highest connection success ratio. If more than one source addresses has the same highest connection success ratio, then we evict the oldest entry, which has the largest value of *TIMER2(s)*, first. If more than one source addresses have the same age, then we select one randomly. By the first rule, IP addresses of scanners, including slow scanners, are likely to stay longer than benign IP addresses due to low connection success ratio. Thus, the first eviction rule enables detection of slow scanners. In the following analysis we assume that the entries corresponding to scanners are not evicted because of the low connection success ratio. The effect of each rule in the eviction policy will be investigated in more detail in the future.

IV. PERFORMANCE ANALYSIS CONSIDERING COLLISIONS IN BLOOM FILTERS

In this section, we show that our detection scheme provides guaranteed performance in terms of false positive and false negative probabilities if scanners and benign hosts exhibit different behavior in terms of connection success ratio. In addition, we investigate how many bits are required for Bloom filters in order to guarantee the false positive and false negative probabilities. Table 1 summarizes the major parameters and variables.

Table 1. Major parameters and variables

n	Number of attempts required for detection decision
θ_1	Maximum connection attempt success ratio of scanners
θ_2	Minimum success ratio of benign hosts
δ_1	False negative requirement
δ_2	False positive requirement
η	Detection threshold of connection success ratio
Y_1	Number of responses to a scanner monitored until the number of attempts reaches n
Y_2	Number of responses to a benign source monitored until the number of attempts reaches n
m	Bit vector size of a Bloom filter
k	Number of hash functions used for Bloom filters
p_c	Collision probability of a Bloom filter

We use the statistic of connection success ratio in order to detect scanners. Let θ_1 and θ_2 denote the maximum connection success ratio of scanners and the minimum connection success ratio of benign hosts, respectively. Based on the data analysis result in [9], we assume that $\theta_1 < \theta_2$. Let Y_1 denote the number of distinct destinations which responded to a scanner (s_1) until the number of addresses which are registered in the Bloom filter $\boldsymbol{V_1}(s_1)$ reaches a pre-specified threshold n. Y_2 denotes the number of distinct hosts which responded to a benign source (s_2) until the number of addresses registered in $\boldsymbol{V_1}(s_2)$ reaches n, that is, until the value of $ATTEMPT(s_2)$ becomes n. Then, the requirements on the false positive and false negative probabilities can be expressed as:

- Low false negatives: $\Pr(Y_1 / n > \eta) < \delta_1$, (3)
- Low false positives: $\Pr(Y_2 / n \le \eta) < \delta_2$, (4)

where δ_1 and δ_2 are the user-specified performance requirements on false negatives and false positives, respectively. Parameters, n and η, can be adjusted to satisfy the above two performance requirements. We need to keep the bit vector size m as small as possible for efficient memory usage. We also need to keep n as small as possible in order to detect scanners rapidly with a small number of connection attempts. The minimum value of m and n may not be achieved simultaneously. We attempt to minimize m while keeping n as small as possible. In this section, we find the set of parameters (n, η) which minimize m while satisfying both of the false positive and false negative conditions and keeping the value of n small. The minimum value of m is found as well.

A. *Collision Probabilities of Bloom Filters*

Let m and k be the number of bits allocated for the bit vector of the Bloom filter and the number of hash functions used for the Bloom filter, respectively. After inserting (n-1) destination addresses into a bit vector of size m, the probability that a particular bit is still 0 is $(1-1/m)^{k(n-1)}$. The collision of Bloom filters occurs when a new destination address sees 1 in every bit position indicated by k hash functions. The collision probability of the arriving destination addresses after the (n-1)-th registered address is then

$$(1-(1-1/m)^{k(n-1)})^k \approx (1-e^{-k(n-1)/m})^k. \quad (5)$$

The right hand side of (5) is minimized when $k = \ln 2 \times m/(n-1)$. It then becomes

$$(1/2)^k = (0.6185)^{m/(n-1)} \quad (6)$$

From both (5) and (6), we know that the collision probability decreases as $m/(n-1)$ increases. We also find that if n is increased for a given k, then m also should be increased in order to keep the collision probability at the same level. Thus, the value of n needs to be small in order to reduce the memory size.

B. False Positive and False Negative Probabilities in the Presence of Collisions in Bloom Filters

We now analyze the effect of collisions in Bloom filters on the false positive and false negative probabilities. The collision occurs when a packet corresponding to a new source/destination pair (s, d) arrives and finds that every value of bits corresponding to the hash values of d is set to 1. If collisions occur in both $\boldsymbol{V_1(s)}$ and $\boldsymbol{V_2(s)}$, *ATTEMPT(s)* is not increased since it is not considered a new attempt. Even though a response packet from d to s is observed, *RESPONSE(s)* is not increased either because the address d already exists in the bit vector $\boldsymbol{V_2(s)}$. Thus, the source/destination pair (s, d) that experiences collisions in both $\boldsymbol{V_1(s)}$ and $\boldsymbol{V_2(s)}$ does not affect either *ATTEMPT(s)* or *RESPONSE(s)*.

If a new source/destination pair (s, d) experiences collision in $\boldsymbol{V_1(s)}$ but not in $\boldsymbol{V_2(s)}$, then *ATTEMPT(s)* is not increased since it is not considered as a new attempt of s. But, if there is a response from d to s, then the *RESPONSE(s)* is increased by 1. Since the new attempt is not counted and only the response is counted in this case, this kind of source/destination pairs tend to increase the success ratio compared to the case with no collision. If s is a benign source, then the collision only in $\boldsymbol{V_1(s)}$ is likely to decrease the false positive ratio. If s is a scanner, then this kind of collision is likely to increase the false negative ratio. Thus, we investigate the false negative probability of a malicious source in more detail when such collisions occur.

The first address to be registered in the Bloom filter does not experience collision. As the number of the registered addresses in the Bloom filter increases, the new destination address of s tends to experience higher collision probability as shown in (5). After n-1 addresses are registered, the collision probability is highest at $(1-e^{-k(n-1)/m})^k$. p_c denotes the highest collision probability, i.e.

$$p_c = (1-e^{-k(n-1)/m})^k. \tag{7}$$

As a conservative approximation to obtain an upper bound of the false negative probability, we assume that every new connection attempt of a given source s experiences the same collision probability of p_c except the first attempt which experiences no collision. We now define the following property of binomial distribution.

Lemma 1. Let $X(n,r)$ denote a binomial random variable with n trials and the success probability of r. Then, for r_1, r_2, r_3, and r_4 ($0 \le r_1 \le r_2 < \eta < r_3 \le r_4 \le 1$), we have

$$\Pr(X(n,r_1)/n > \eta) \le \Pr(X(n,r_2)/n > \eta), \tag{8}$$

$$\Pr(X(n,r_4)/n \le \eta) \le \Pr(X(n,r_3)/n \le \eta). \tag{9}$$

Proof: The proof is given in Appendix A.

Due to the collision in Bloom filters, it is likely that more than n attempts are required to fill the Bloom filter which accommodates only n addresses. Let A denote the total number of connection attempts that a source address s makes until the n-th connection attempt is counted by the Bloom filter. $Y_1(j)$ and $Y_2(j)$ denote the conditional random variables $Y_1 \mid A = j$ and $Y_2 \mid A = j$, respectively. Let $Y_1'(n)$ denote $Y_1(n)$ when the scanner has the highest success ratio of θ_1. Let $Y_2'(n)$ denote $Y_2(n)$ when the benign source has the lowest success ratio of θ_2. We assume that the response result of each attempt is independent and identically distributed (i.i.d.) for the same source address. We then prove the following about the false positive and false negative probabilities.

Proposition 1. If a set of (n, η) satisfy (10) and (11), then the false positive and false negative probability requirements of (3) and (4) are satisfied.

$$\Pr(Y_1'(n) > n\eta) + 1 - (1-p_c)^{n-1} < \delta_1, \tag{10}$$

$$\Pr(Y_2'(n) \le n\eta) < \delta_2, \tag{11}$$

where $Y_1'(n) \sim \text{Binomial}(n,\theta_1)$ and $Y_2'(n) \sim \text{Binomial}(n,\theta_2)$.

Proof: The proof is given in Appendix B.

We now solve (10) and (11) simultaneously to find the value of n and η which guarantee the false positive and false negative probabilities. In order to solve the inequality (10) explicitly in terms of n, we consider the following set of decomposed inequalities:

$$\Pr(Y_1'(n) > n\eta) < x, \tag{12}$$

$$1-(1-p_c)^{n-1} \le \delta_1 - x, \tag{13}$$

where $0 < x < \delta_1$. If we find the set of (n, η) which satisfies both (12) and (13) simultaneously, then those values of (n, η) also satisfy (10).

Thus, if we show that there exists n and η that satisfy (11), (12), and (13) simultaneously, then it proves that our scheme guarantees the false positive and false negative probabilities even in the presence of collisions in Bloom filters. Because it is not easy to obtain the range of n which satisfies (11) and (12) in a closed form with the exact binomial distribution, we first consider a simple form of upper bound for binomial distribution. For a binomial random variable X ~ Binomial(n, p), the following inequalities can be derived using Chernoff bounds [20]:

$$\Pr(X - np \ge \xi) \le \exp(-2\xi^2 / n),$$

$$\Pr(X - np \le -\xi) \le \exp(-2\xi^2 / n).$$

If we apply the above inequalities, we then obtain

$$\Pr(Y_1'(n) > n\eta) \le \exp\{-2n(\eta - \theta_1)^2\}, \tag{14}$$

$$\Pr(Y_2'(n) \le n\eta) \le \exp\{-2n(\theta_2 - \eta)^2\}. \tag{15}$$

If we consider (12) and (14) simultaneously, then n that satisfies

$$n > \frac{-\ln x}{2(\eta - \theta_1)^2} \tag{16}$$

also satisfies (12). If we consider (11) and (15) simultaneously, then n that satisfies

$$n > \frac{-\ln \delta_2}{2(\theta_2 - \eta)^2} \tag{17}$$

also satisfies (11). Let $h_1'(\eta)$ and $h_2'(\eta)$ denote the lower bounds on the right hand side of (16) and (17), respectively. (13) leads to

$$n \le \frac{\ln(1-\delta_1+x)}{\ln(1-p_c)} + 1. \tag{18}$$

Let $h_3'(m/(n-1))$ denote the right hand side term of (18). The set of (n, η) that satisfies (16), (17), and (18) simultaneously is expressed as a shaded area in the Fig. 2. Let η^* and n^* denote the value of η and n that is on the intersection of $h_1'(\eta)$ and $h_2'(\eta)$. From Fig. 2, we find that the minimum value of n is obtained when $\eta = \eta^*$. The value of η^* is then

$$\eta^* = \frac{\theta_2 + \sqrt{\ln \delta_2 / \ln x}\,\theta_1}{1 + \sqrt{\ln \delta_2 / \ln x}}.$$

The minimum value of n is determined as $\lfloor n^* \rfloor + 1$, where $\lfloor u \rfloor$ denotes the largest integer that is smaller than or equal to u.

The required size of bit vector m and the number of hash functions k can be determined by (18). We assume that the value of m/(n-1) is preserved by adaptively changing the value of m according to the value of n. In order to assure that $(\eta^*, \lfloor n^* \rfloor + 1)$ exists in the shaded area of Fig. 2, $n = \lfloor n^* \rfloor + 1$ has to satisfy (18). We substitute $\lfloor n^* \rfloor + 1$ for n and solve it in terms of p_c. We then have

$$p_c \le 1-(1-\delta_1+x)^{1/\lfloor n^* \rfloor}. \tag{19}$$

Since the minimum value of p_c is $0.6185^{m/(n-1)}$ from (5), (6), and (7), the range of $m/(n\text{-}1)$ is given by (19) as

$$\frac{m}{n-1} \ge \frac{\ln\{1-(1-\delta_1+x)^{1/\lfloor n^* \rfloor}\}}{\ln 0.6185}.$$

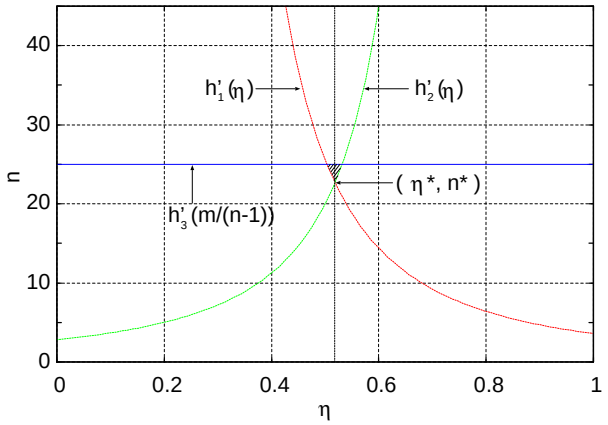

Fig. 2. Solution area for (16), (17), and (18)

The value of m is minimized when n has the minimum value of $\lfloor n^* \rfloor+1$ and the minimum value of m is determined as

$$m^* = \left\lceil \frac{\ln\{1-(1-\delta_1+x)^{1/\lfloor n^* \rfloor}\}}{\ln 0.6185} \right\rceil \lfloor n^* \rfloor, \tag{20}$$

where $\lceil u \rceil$ denotes the smallest integer that is larger than or equal to u. The value of corresponding k is determined by $k = \ln 2 \times m^* / \lfloor n^* \rfloor$ from (6) and (20). We find a set of parameters (n, η, m, k) for any given constraints δ_1 and δ_2, and we thus prove that our scheme guarantees false positive and false negative probabilities even in the presence of collision in Bloom filters.

C. Required Memory Size

We now show the required memory size for the required performance in terms of false positive and false negative probabilities. We find the set of parameters (n, η, m, k) which can minimize the required memory size for given constraints, δ_1 and δ_2.

In the previous subsection, we show the parameter selection process for an arbitrary value of x ($0 < x < \delta_1$) in (12) and (13). In order to find the minimum value of m, we need to perform the same parameter selection process for various values of x in the interval of $(0, \delta_1)$. We find the minimum bit vector size m as follows:

- For each value of x in $(0, \delta_1)$, we find the set of parameters (n, η, m, k) according to the given selection process. We then minimize the value of n and m while satisfying the false positive and false negative probabilities.
- In the next step, we select the set (n, η, m, k) which minimizes m after obtaining a list of the set (n, η, m, k) for various values of x in $(0, \delta_1)$.

Fig. 3. shows the values of m computed using the above procedure for various values of x ($0 < x < \delta_1$) when $\theta_1 = 0.2$, $\theta_2 = 0.8$, and $\delta_1 = \delta_2 = 0.05$. The value of x changes from 0.0005 to 0.0495 ($= 0.99\delta_1$) with an

increment of 0.0005. Since the value of m takes discrete values from a very limited set of integers as shown in Fig. 3, we find a minimum value of m through the exhaustive search. In Figure 3, the minimum value of m (252) is obtained at $x = 0.025$. Although 252 is the minimum value of m when the Chernoff bound is used, Chernoff bound is a rather loose bound. If we use the binomial distribution for $Y_1'(n)$ in (12) and $Y_2'(n)$ in (11) in order to find the minimum value of n that satisfies (12) and (11) simultaneously, then we find smaller value of n. The optimal value of η can not be solved explicitly using the binomial distribution. Thus, we use the value of η obtained from Chernoff bound-based scheme to compute the near-optimal value of n with binomial distributions assuming that the optimal value of η for the binomial distribution is not much different from the value obtained from Chernoff bound-based scheme.

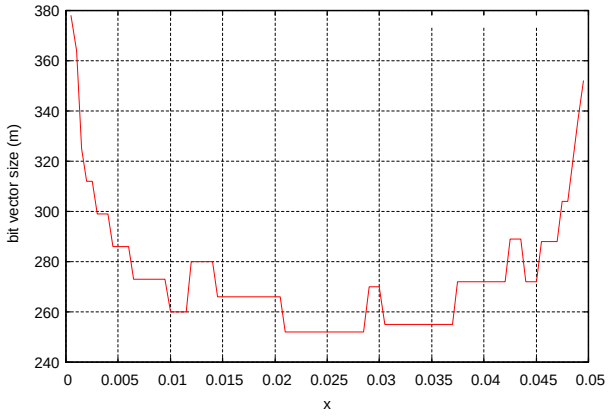

Fig. 3. Bit vector size (m) for various values of x

Table 2 shows the set of parameters (n, η, m, k) that minimize the value of m under the given conditions of ($\theta_1, \theta_2, \delta_1, \delta_2$) when Chernoff bound is used. Table 3 shows the result when the binomial distribution is used. The parameters in Table 3 guarantee the required false positive and false negative probabilities. Comparing the results in Table 2 and 3, we find that the values of n and m obtained from the Chernoff bound are always upper bound of the results obtained from binomial distribution. The value of n is kept smaller than 25 for every case in Table 3.

Fig. 1 shows that $54 + 2m$ (=$32 + 2m + 5 + 5 + 4 + 8$) bits are required for each source address. For one million source addresses, ($54+2m$) Mbits of SRAM memory is required to implement our scheme. For example, when $\theta_1 = 0.1$, $\theta_2 = 0.9$, and $\delta_1 = \delta_2 = 0.05$, 20 bits are required for m as shown in Table 3. In this case, the total memory requirement is 94 Mbits. This can be implemented with a 16 MB SRAM. Table 3 shows that more strict performance can be provided in terms of false positive and false negative probabilities as the memory is increased.

Table 2. Parameters which minimize m when Chernoff bound is used

Θ_1	Θ_2	δ_1	δ_2	η	n	m	k
0.2	0.8	0.05	0.05	0.516	19	252	10
0.2	0.8	0.01	0.01	0.511	28	486	13
0.2	0.8	0.005	0.005	0.508	31	600	14
0.2	0.8	0.001	0.001	0.506	40	936	17
0.1	0.9	0.05	0.05	0.528	11	120	9
0.1	0.9	0.01	0.01	0.508	15	252	13
0.1	0.9	0.005	0.005	0.516	18	306	13
0.1	0.9	0.001	0.001	0.512	23	484	16

Table 3. Parameters which minimize m when Binomial distribution is directly used

Θ_1	Θ_2	δ_1	δ_2	η	n	m	k
0.2	0.8	0.05	0.05	0.516	7	78	10
0.2	0.8	0.01	0.01	0.511	13	216	13
0.2	0.8	0.005	0.005	0.508	15	294	15
0.2	0.8	0.001	0.001	0.506	24	506	16
0.1	0.9	0.05	0.05	0.528	3	20	7
0.1	0.9	0.01	0.01	0.508	5	68	12
0.1	0.9	0.005	0.005	0.516	7	102	12
0.1	0.9	0.001	0.001	0.512	9	192	17

V. SAMPLING-BASED EXTENSION

In this section, we introduce a sampling technique to address a short history-based scanner detection scheme. If a scanner detection scheme makes decisions based on a small number of connection attempts, then it is possible to evade the detection system with only a small number of known IP addresses.

Up to this point, a scanner is assumed to select target IP addresses randomly without knowing any valid IP addresses in advance. However, more advanced worms and bots use valid IP addresses to accelerate the spreading or evade detection like the hit-list scanning technique [13, 14]. In this section, we extend our detection scheme with a sampling technique and show that our sampling-based extended scheme can detect scanners with a fixed number of valid IP addresses while guaranteeing false positive and false negative probabilities.

We explain the shortcoming of the current version of our detection scheme with an example. Knowing the value of n, the scanner s with n valid IP addresses can operate without being detected by our current scheme. Let us assume that the detection threshold η is 0.5 and n is fixed to 10. Let d_1, …, d_{10} denote the valid IP addresses that the scanner s already knows. Let a_1, a_2, … denote the random addresses to be scanned. If the scanner scans with 6 of valid IP addresses among $d_1 \sim d_{10}$, and 4 random addresses from a_1, a_2, …, then the success ratio would be at least 0.6 every time. With the success ratio of 0.6, the scanner is not detected. This kind of problem also has been noted in [10]. We combine a sampling technique with our scheme to detect such scanners. We randomly sample source/destination pairs with a probability of 0.1 and consider the connection attempts of only those sampled source/destination pairs. Then, on average, only 1 destination among $d_1 \sim d_{10}$ is likely to be considered in the connection success ratio. Thus, the effect of pre-acquired valid IP addresses on the connection success ratio can be controlled by the sampling probability. We use the sampling technique suggested in [4] to sample source/destination pairs. We use a uniform random hash function h_m to map (s, d) pairs to [0,1). If the random number allocated to (s, d) is less than the sampling probability p_s, then the (s, d) pair is sampled and the connection status is monitored[3].

For the performance analysis, we assume that the number of valid IP addresses known to the scanner is limited to l. We also assume that the scanner continues to scan after the scanning of valid IP addresses. We then show that the following statement is valid.

Proposition 2. For x_1 and x_2 ($x_1 > 0, x_2 > 0, x_1 + x_2 < \delta_1$), if the 5-tuple of (n, η, m, k, p_s) satisfy the following relations,

$$1-(1-p_c)^n \le x_1, \tag{21}$$

$$\sum_{n \le i < \alpha l} \binom{i-1}{n-1} p_s^n (1-p_s)^{i-n} \le x_2, \tag{22}$$

$$\Pr(\hat{Y}_1(n, \alpha l) > n\eta) < \delta_1 - x_1 - x_2, \tag{23}$$

$$\Pr(Y_2'(n) \le n\eta) < \delta_2, \tag{24}$$

[3] Since the sampling is based on (s, d) pair and the attempt and the response are counted at most one time for each (s, d) pair, duplicate attempts to the same valid IP address can not affect the connection success ratio of s.

where p_s is a source/destination pair sampling probability, α is a real number larger than $(1-\theta_1)/(\theta_2-\theta_1)$, and $\hat{Y}_1(n,\alpha l) \sim \text{Binomial}(n, \theta_1+(1-\theta_1)/\alpha)$. Then the false positive and false negative probability requirements of (3) and (4) are satisfied.

Proof: The proof is given in Appendix C.

(21), (23) and (24) have the same form as (13), (12) and (11), respectively, although the success probability is increased by $(1-\theta_1)/\alpha$ in case of Y_1 and the power of $1-p_c$ is increased from n-1 to n. Thus, the set of (n, η, m, k) can be obtained from (21), (23), and (24) in the similar way as in Section III. The value of α should be larger than $(1-\theta_1)/(\theta_2-\theta_1)$ in order to prevent the increased success probability of the scanner, $(\theta_1+(1-\theta_1)/\alpha)$, from being larger than the minimum success probability of benign hosts, θ_2. If $\alpha > (1-\theta_1)/(\theta_2-\theta_1)$, then the false positive and false negative probability requirements are guaranteed. We now study the effect of α.

We induce the property of α from (22). If we let A_s denote the number of attempts required to sample n attempts with no collision in Bloom filter, i.e. $p_c = 0$, then (22) becomes

$$\Pr(A_s < \alpha l) \le x_2. \tag{25}$$

From (22), the expectation of A_s is given by $E[A_s] = n/p_s$. From the Markov inequality [21], we obtain

$$\Pr(A_s < \alpha l) \ge 1 - E[A_s]/(\alpha l). \tag{26}$$

Combining (25) and (26) yields $E[A_s] \ge \alpha l(1-x_2) \approx \alpha l$, since x_2 is usually negligibly small compared with 1. If we consider the collision in Bloom filters, then $E[A] = n/(p_s(1-p_c)) \ge E[A_s]$ and we have

$$E[A] \ge \alpha l. \tag{27}$$

If we combine $E[A_s] = n/p_s$ and $E[A_s] \ge \alpha l$, we get

$$p_s \le n/\alpha l. \tag{28}$$

If we increase α, then the effective success ratio of the scanner $(\theta_1+(1-\theta_1)/\alpha)$ decreases. α thus decreases the effect of scanners with known valid IP addresses. As the difference between the success ratios of scanners and benign hosts increases, our detection system operates faster with smaller number of connection attempts (n) and the smaller memory size. However, the sampling increases the time to detect scanners. (27) shows that a large value of α tend to increase data collection time. As α increases, the memory size is decreased, but the data collection time is increased due to the decrease of the sampling probability in (28).

The value of sampling probability p_s can be computed from (22) when the value of n is determined. If we focus on the minimization of the memory size or bit vector size (m), then the optimal case is likely to be achieved when x_2 is negligibly small. If the value of x_2 is very small, then p_s should be small according to (22). The small value of p_s leads to long data collection time. In order to avoid too small value of p_s, we add the constraint that $x_2 = 0.1x_1$ to (21), (22), and (23).

Table 4 shows the set of parameters (n, η, m, k, p_s) obtained from (21) ~ (24) when Chernoff bound is used to approximate Binomial distributions. Table 5 shows the set of parameters (n, η, m, k, p_s) obtained from (21) ~ (24) when Binomial distribution is used without approximation. Table 4 and 5 shows that the resulting parameters except p_s are not changed even though the number of known addresses l is changed. Only p_s is dependent on l as shown in (22). If we compare the results of Table 5 with the same cases in Table 3, the values of the parameters (n, m, k) are identical especially when $\alpha = 20$. Thus, the cost of adding the sampling to our detection scheme is not significant. Tables 4 and 5 show that the large α tends to decrease the required memory size, but n/p_s ($\approx E[A]$) increases as α increases. The value of α can be determined considering the relative importance between memory budget and short data collection time.

Table 4. Parameters which minimize m when sampling technique and Chernoff bound are used ($\theta_1 = 0.2, \theta_2 = 0.8$)

$\delta_1(=\delta_2)$	α	l	η	n	m	k	p_s
0.05	10	10	0.550	24	345	11	0.13047
		1000	0.550	24	345	11	0.00122
	20	10	0.539	22	294	10	0.05764
		1000	0.539	22	294	10	0.00056
0.005	10	10	0.546	41	840	15	0.24621
		1000	0.546	41	840	15	0.00220
	20	10	0.530	37	720	14	0.10206
		1000	0.530	37	720	14	0.00097

Table 5. Parameters which minimize m when sampling technique and Binomial distribution are used ($\theta_1 = 0.2, \theta_2 = 0.8$)

$\delta_1(=\delta_2)$	α	l	η	n	m	k	p_s
0.05	10	10	0.550	7	78	10	0.01687
		1000	0.550	7	78	10	0.00016
	20	10	0.539	7	78	10	0.00830
		1000	0.539	7	78	10	0.00008
0.005	10	10	0.546	18	306	13	0.07380
		1000	0.546	18	306	13	0.00069
	20	10	0.530	15	294	15	0.02301
		1000	0.530	15	294	15	0.00022

Additional benefit of the sampling is that we can cover more connection attempts with a limited memory size. Let us assume that the memory space can accommodate only M source/destination pairs. Without sampling, we can track only M source/destination pairs. But with sampling with probability of p_s, an average of M/p_s (s, d) pairs can be tracked. For $p_s = 0.1$, we can cover 10 times more (s, d) pairs with the same memory.

If we fix the sampling probability to a very small number in order to detect scanners with large l, then it takes long time to detect scanners with small l. We alleviate this problem by running two detection systems in parallel with different sampling probabilities. If a scanner finishes probing just after scanning the known l addresses, the success ratio of the scanner would remain near 1.0. This kind of scanners with high success ratio can not be detected based on the attempt success ratio. Our extended detection system can detect a scanner with a large l only when the real success ratio of the scanner drops sufficiently lower than θ_2 by scanning random IP addresses.

VI. NUNERICAL RESULTS

In this section, we evaluate the performance of our scheme by simulation. We use packet traces taken from NLANR archive [22] as the base traffic. We also inject packet traces of 1000 normal hosts with a success ratio of 0.8 (θ_2) and 1000 malicious hosts with a success ratio of 0.2 (θ_1). We model a successful attempt with a single pair of bidirectional packets. The response time of the response packet is distributed uniformly in the interval of [0, 450] *msec*. The interval between successive attempts is modeled with an exponential distribution. The base traces from NLANR are described in Table 6.

Table 6. Description of NLANR traces

Trace	Duration (sec)	No. packets	No. distinct sources	No. distinct source-destination pairs
1	6720	3496843	752617	1166423
2	12900	2116553	536554	810486
3	4615	1837565	445712	650009

We limited the number of total entries to 0.5 million. The number of addresses in the connection status table is 125,000 and there are 4 entries at each hash value. MD5 is used for hash functions. Table 7 shows the simulation results for several scenarios. For the scanner without any valid IP addresses in advance, i.e. l = 0, we use the

parameters given in Table 3 for our detection system. For the scanner with 10 valid IP addresses in advance, i.e. l = 10, the sampling-based extended scheme is applied with the parameters corresponding to α = 10 in Table 5.

For l = 0, we observe that the false positive probabilities are guaranteed. The false negative probability is guaranteed in scenarios 5 and 6 for trace 3, but it is not guaranteed in scenarios 1~4 for traces 1 and 2. When we compute the false positives, we include benign traffic in NLANR traces as well as the injected packets. Although the injected connection attempts of normal behavior have a success ratio of 0.8, each IP address in the real trace does not make many connection attempts to distinct hosts and has a rather high success ratio. Since the number of sources corresponding to this normal behavior in the original trace is much larger than the number of injected source addresses, the false positive probability tends to be very small. In Section IV and V, we assume that the memory is not overloaded and the entries corresponding to scanners are not evicted due to the low connection success ratio. If we compare the number of distinct sources of traces 1, 2, and 3 with the total number of entries in the memory, 0.5 million, the number of distinct sources is larger than 0.5 million for traces 1 and 2. We find that the NLANR traces also contain some hosts with a rather low connection success ratio of below 0.8 and they are competing for memory space with injected scanners causing eviction of entries of scanners sometimes. Since this kind of contention is severe especially when the memory is overloaded, the resulting eviction events increase the false negative probability for the overloaded cases (traces 1 and 2). For example, in the scenarios 1, 3 and 5 the decision is made based on 7 attempts according to Table 3. Thus, we have each scanner send only 10 attempt packets with a low rate during the whole simulation period. If the entry corresponding to this scanner is evicted after the arrival of 4-th attempt due to temporal overload on the memory, then the remaining 6 attempts are not sufficient to make a decision and this scanner may not be detected. However, even for the scenarios 1 and 2, the obtained false positive probability is still close to the target value. We note that only 10 and 25 scanning packets have been sent by a scanner in scenarios 5 and 6, respectively. Thus, the scanning rate is as low as 1 packet per several hundred seconds. However, our scheme detects those slow scanners with guaranteed performance in those scenarios when the memory is not heavily overloaded.

In scenarios 7~10 with scanners using 10 valid IP addresses in advance, we use sampling-based scheme described in Section V. The environment is similar to that of scenario 1~4, but each scanner sends 1.5 times more packets than the number required to fill Bloom filters under sampling. We observe that the false positive and false negative probabilities are satisfied in these scenarios. Since sampling has the effect of reducing load to the memory by sampling probability, the number of evictions is decreased significantly as shown in Table 7. The false positive and false negative probabilities are guaranteed according to the Proposition 2 in Section V. Although the sampling technique is incorporated to detect more intelligent scanners with valid IP address information, we also find that the sampling is useful to resolve the overload problem on memory as shown by results for scenarios 1~4 and those for 7~10.

Table 7. Simulation results ($\theta_1 = 0.2, \theta_2 = 0.8$)

Scenario	Known addresses (l)	Trace	δ_1/δ_2	False positives	False negatives	No. evictions
1	0	1	0.05/0.05	3.64e-5	0.070	306210
2	0	1	0.005/0.005	2.68e-6	0.008	307766
3	0	2	0.05/0.05	5.50e-5	0.053	130214
4	0	2	0.005/0.005	3.75e-6	0.006	131037
5	0	3	0.05/0.05	6.82e-5	0.035	73632
6	0	3	0.005/0.005	1.13e-5	0.004	74091
7	10	1	0.05/0.05	9.29e-5	0.013	1
8	10	1	0.005/0.005	7.97e-6	0	209
9	10	2	0.05/0.05	1.00e-4	0.020	0
10	10	2	0.005/0.005	1.49e-5	0.001	13

VII. CONCLUSIONS

A new scanning detection scheme is proposed based on the connection attempt success ratio. The proposed scheme can detect scanners, including slow scanners, with guaranteed false positive and false negative

probabilities under a memory-limited environment. Since a high priority is given to source addresses with a low connection success ratio, the entries corresponding to scanners can be retained in the connection status table longer than those of benign IP addresses, and thus, detection of slow scanners is possible in our scheme. The detection threshold, the required memory size, and other system parameters are analytically derived in order to guarantee false positive and false negative probabilities.

Usually detection of slow scanners require management of per-flow states with the $O(N^2)$ of memory [7, 11], but our scheme reduces the memory requirement to $O(N)$ by making decisions with a fixed number of connection attempts. We used Bloom filters to further reduce the memory size. Our scheme can detect scanners with a rather small number of connection attempts, less than 25, and the complexity of the proposed scheme is sufficiently low to allow monitoring at high speed links. By applying a sampling technique to our detection algorithm, we can detect more intelligent scanners that use pre-acquired valid IP addresses while maintaining the guaranteed false positive and false negative probabilities. We evaluate the performance of our schemes by simulation and find that the fast detection and detection of slow scanners are achieved while maintaining both the false positive and the false negative probabilities.

REFERENCES

[1] CERT Coordination Center, http://www.cert.org/stats/cert_stats.html

[2] T. Holz, "A short visit to the bot zoo," IEEE Security & Privacy Magazine, vol. 3, no. 3, pp. 76-79, May-June 2005.

[3] N. Weaver, S. Staniford, and V. Paxson, "Very fast containment of scanning worms," in *Proc. The 13-th Usenix Security Conference*, 2004.

[4] S. Venkataraman, D. Song, P. B. Gibbons, and A. Blum, "New streaming algorithms for fast detection of superspreaders," in Network and Distributed Systems Security Symposium, Feb. 2005.

[5] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Proc. IEEE Symposium on Research in Security and Privacy*, pp. 296-304, 1990.

[6] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. 13-th Conference on Systems Administration (LISA-99)*, pp. 229-238, Berkeley, CA, Nov. 7-12, 1999. USENIX Association.

[7] V. Paxson, "Bro: a system for detecting network intruders in real-time," Computer Networks, vol. 31, pp. 2435-2463, 1999.

[8] S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo, "Surveilance detection in high bandwidth environments," in *Proc. 2003 DARPA DISCEX III Conference*, pp. 130-139, Washington, DC, 2003. IEEE Press. 22-24, April 2003.

[9] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proc. IEEE Symposium on Security and Privacy*, May 2004.

[10] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," in *Proc. International Symposium on Recent Advances in Intrusion Detection (RAID)*, Sep. 2004.

[11] R. R. Kompella, S. Singh, and G. Varghese, "On scalable attack detection in the network," in Proc. Internet Measurement Conference (IMC), Oct. 2004.

[12] S. Staniford, J. Hoagland, and J. McAlerney, "Practical automated detection of stealthy portscans," Journal of Computer Security, vol. 10, no. 1, pp. 105-136, 2002.

[13] J. Wu, S. Vangala, L. Gao, and K. Kwiat, "An effective architecture and algorithm for detecting worms with various scan techniques," in *Proc. Network and Distributed System Security Symposium*, 2004.

[14] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in your spare time," in *Proc. The 11-th USENIX Security Symposium*, USENIX, Aug. 2002.

[15] W. Fang and L. Peterson, "Inter-as traffic patterns and their implications," in *Proc. IEEE GLOBECOM*, Dec. 1999.

[16] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in *Proc. Internet Measurement Conference (IMC)*, Oct. 2003.

[17] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," Internet Mathematics, vol. 1, no. 4, pp. 485-509, 2003.

[18] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," Techinical Report 1361, Computer Sciences Department, Univ. of Wisconsin-Madison, Feb. 1998.

[19] C. Fraleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," IEEE Network, vol. 17, no. 6, Nov.-Dec. 2003.

[20] M. Harchol-Balter, 15-359 Probability and Computing - Inequalities, http://www.cs.cmu.edu/~15359/Notes/05/l5.ps, 2005.

[21] A. Papoulis, *Probability, random variables, and stochastic processes, 3rd Ed*., New York: McGraw-Hill, 1991.

[22] NLANR. National laboratory for applied network research. http://pma.nlanr.net/Traces/, 2003.

[23] E. R. Dougherty, *Probability and Statistics for the Engineering, Computing and Physical Sciences*, Prentice-Hall, 1990.

APPENDIX A – PROOF OF LEMMA 1

Since n is a fixed number, we have $\Pr(X(n,r)/n > \eta) = \Pr(X(n,r) > n\eta)$ and it can be expressed as

$$\Pr(X(n,r) > n\eta) = \sum_{j=\lfloor n\eta \rfloor+1}^{n} \binom{n}{j} r^j (1-r)^{n-j}.$$

Since n and η are usually fixed in the above equation, we consider the right hand side term as a function of r and let $f(r)$ denote the right hand side term of the above equation. If we differentiate $f(r)$ with respect to r, then we have

$$f'(r) = \sum_{j=\lfloor n\eta \rfloor+1}^{n} \binom{n}{j} r^{j-1}(1-r)^{n-j-1}(j-nr) \tag{A1}$$

Let us consider r less than η, i.e. $r < \eta$. Since $j > n\eta$, we have $j > n\eta > nr$. Thus, $f'(r) > 0$ for $r < \eta$ by (A1) and we have $f(r_1) \le f(r_2)$ since $r_1 \le r_2 < \eta$. Thus, (8) is proved. (9) can also be proved in a similar way. n

APPENDIX B – PROOF OF PROPOSITION 1

A is the total number of connection attempts of a source address s until the n-th connection attempt is counted by the Bloom filter. Since the packet arriving at an empty Bloom filter does not experience collision, the distribution of A is given as

$$\Pr(A = n+i) = \binom{n+i-2}{n-2}(1-p_c)^{n-1} p_c^{\,i}, \tag{B1}$$

where i = 0, 1, 2, …. Let us investigate the distribution of $Y_1(j)$, i.e. $Y_1 \mid A = j$. In real case, the value of *RESPONSE* may not reach j due to collisions in V_2. But, we assume that there is no collision in V_2 and the value of *RESPONSE* can reach up to j as a conservative approximation in terms of false negative probability. This assumption is conservative because large value of *RESPONSE* implies high false negatives for scanners. Let $Z_1(i)$ be a random variable that represents the result of the i-th connection attempt of a malicious source. $Z_1(i)$ is 1 if the i-th attempt is a success, and 0 if the attempt is a failure. Then, $Y_1(j)$ can be expresses in terms of $Z_1(i)$ as

$$Y_1(j) = \sum_{i=1}^{j} Z_1(i), \quad j \ge n.$$

Since we assume that the random variables $Z_1(i)$'s (i = 1, 2, …) are independent and identically distributed (i.i.d.), $Y_1(j)$ has a binomial distribution.

In order to discriminate the hosts with the success ratio of θ_1 from the hosts with the success ratio of θ_2, we need to put the detection threshold η between them. Thus, we assume $\theta_1 < \eta < \theta_2$. The false negative probability of a scanner s_1 can be expressed as

$$\begin{aligned} \Pr(Y_1/n > \eta) &= \sum_{j=n}^{\infty} \Pr(Y_1/n > \eta \mid A = j) \Pr(A = j) \\ &\le \Pr(Y_1/n > \eta \mid A = n) \Pr(A = n) + 1 - \Pr(A = n). \end{aligned} \tag{B2}$$

By the definition of $Y_1(n)$, we have $\Pr(Y_1/n > \eta \mid A = n) = \Pr(Y_1(n)/n > \eta)$. Since $\Pr(A = n) = (1-p_c)^{n-1}$ by (B1), (B2) can be changed into

$$\Pr(Y_1/n > \eta) \le \Pr(Y_1(n)/n > \eta)(1-p_c)^{n-1} + 1 - (1-p_c)^{n-1}.$$

Since we can obtain $\Pr(Y_1'(n) > n\eta) \ge \Pr(Y_1(n) > n\eta)$ from Lemma 1 and $(1-p_c)^{n-1} \le 1$ to the right hand side of the above inequality, we have

$$\Pr(Y_1/n > \eta) \le \Pr(Y_1'(n) > n\eta) + 1 - (1-p_c)^{n-1}. \tag{B3}$$

Thus, if we find the set of (n, η) which satisfies (10), then for those set of (n, η) the false negative probability is guaranteed by (B3) and (10).

Y_2 is the number of connection successes of a benign host observed until n attempts are counted by the Bloom filter. Then, we have

$$\Pr(\frac{Y_2}{n} \le \eta) = \sum_{j=n}^{\infty} \Pr(\frac{Y_2}{n} \le \eta \mid A = j) \Pr(A = j). \quad \text{(B4)}$$

In this case, we do not assume that there is no collision in V_2 since this assumption leads to decrease of the above false positive probability optimistically. Let R denote the maximum number the *RESPONSE* value can reach in this case. We consider only the attempts which can contribute to the *RESPONSE* value without collision in the Bloom filter V_2 in Y_2. Then, we have

$$\Pr(\frac{Y_2}{n} \le \eta \mid A = j) = \sum_{i=n}^{j} \Pr(\frac{Y_2}{n} \le \eta \mid R = i, A = j) \Pr(R = i \mid A = j). \quad \text{(B5)}$$

Since we assumed that the outcomes of the attempts of a benign host are i.i.d., we have

$$\Pr(Y_2 \le n\eta \mid R = i, A = j) = \Pr(\tilde{Y}_2(i) \le n\eta). \quad \text{(B6)}$$

where $\tilde{Y}_2(i)$ denotes $Y_2 \mid R = i, A = j$ and $\tilde{Y}_2(i) \sim \text{Binomial}(i, p_2)$, where p_2 is the success probability of the benign source. We can easily show that $\Pr(\tilde{Y}_2(i) \le n\eta)$ is a non-increasing function with respect to i when $i \ge n$ and $p_2 \ge \theta_2 > \eta$, that is,

$$\Pr(\tilde{Y}_2(n) \le n\eta) \ge \Pr(\tilde{Y}_2(i) \le n\eta), \quad i \ge n, \quad \text{(B7)}$$

Then, from (B4), (B5), (B6), and (B7), we can obtain the following bound:

$$\Pr(\frac{Y_2}{n} \le \eta) \le \Pr(\tilde{Y}_2(n) \le n\eta). \quad \text{(B8)}$$

When A=n, there is no collision in the Bloom filter and R is also equal to n. Thus, we have $\Pr(\tilde{Y}_2(n) \le n\eta) = \Pr(Y_2(n) \le n\eta)$ from (B5), where $Y_2(n)$ denotes $Y_2 \mid A = n$, and (B8) can be changed into

$$\Pr(\frac{Y_2}{n} \le \eta) \le \Pr(Y_2(n) \le n\eta). \quad \text{(B9)}$$

Since $\Pr(Y_2'(n) \le n\eta) \ge \Pr(Y_2(n) \le n\eta)$ by Lemma 1, from (B9) we have

$$\Pr(Y_2 / n \le \eta) \le \Pr(Y_2'(n) \le n\eta). \quad \text{(B10)}$$

By (B10), the false positive probability is guaranteed if (11) is satisfied. n

APPENDIX C – PROOF OF PROPOSITION 2

As the scanner tries more known IP addresses before the sampling is over, the monitored success probability is likely to increase. Thus, the connection success ratio can be maximized only when the scanner tries as many known IP addresses as possible before the sampling is over. The false negative probability is also maximized when the scanner tries as many valid IP addresses as possible. If we can guarantee the false negative probability in this worst case, the false negative probability should be guaranteed in other cases, i.e. when the scanner tries less number of known IP addresses, too. Thus, we assume that scanners attempts to scan known IP addresses as early as possible for a high connection success ratio.

Let A denote the number of connection attempts made until the *ATTEMPT* value of a given source address reaches n. Let B denote the number of sampled attempts until the *ATTEMPT* value of the source address reaches n.

As a conservative approximation we assume that every sampled attempt or (s, d) pair experience the collision probability of p_c given by (7) in Bloom filters. An attempt can be registered into a Bloom filter if it is sampled and does not experience collision in Bloom filters. Thus, the distribution of A is given by

$$\Pr(A=i)=\binom{i-1}{n-1}(p_s(1-p_c))^n(1-p_s(1-p_c))^{i-n}, \tag{C1}$$

where i = n, n+1, ….

By conditioning on A, we can obtain the following relation regarding false negative probability:

$$\begin{aligned}\Pr(\frac{Y_1}{n}>\eta)&=\sum_{i=n}^{\infty}\Pr(\frac{Y_1}{n}>\eta \mid A=i)\Pr(A=i)\\ &\le \Pr(A<\alpha l)+\sum_{i\ge\alpha l}\Pr(\frac{Y_1}{n}>\eta \mid A=i)\Pr(A=i).\end{aligned} \tag{C2}$$

By conditioning on B, we can obtain

$$\Pr(\frac{Y_1}{n}>\eta \mid A=i)=\sum_{j=n}^{i}\Pr(\frac{Y_1}{n}>\eta \mid B=j, A=i)\Pr(B=j \mid A=i). \tag{C3}$$

In order to obtain an upper bound of a false negative probability, if we assume that the responses of the sampled attempts does not experience collisions in V_2 of the corresponding source address as a conservative approximation, then the *RESPONSE* value can reach up to j when B=j. Let us consider the case of $i > l$ since $i \ge \alpha l$ in (C2) and $\alpha > 1$ by the assumption. Since we assumed that the scanner attempts to scan l known IP addresses first, those attempts to l known IP addresses are included in the total number of attempts i. If the sampling probability is fixed to p_s, then whether a specific attempt among i attempts is sampled or not is independent of sampling of other attempts by our sampling scheme. Then, the probability that the first sampled attempt is from the list of known IP addresses of a scanner is l/i. Before evaluating (C3), let us investigate the distribution of $Y_1 \mid B=n, A=i$. Let C denote the number of attempts to known addresses among B sampled attempts. Then, the distribution of $Y_1 \mid B=n, A=i$ can be expressed as

$$\Pr(Y_1=z \mid B=n, A=i)=\sum_{k=0}^{n}\Pr(Y_1=z \mid C=k, B=n, A=i)\Pr(C=k \mid B=n, A=i). \tag{C4}$$

Although $\Pr(C=k \mid B=n, A=i)$ is given as a hypergeometric distribution of $\binom{l}{k}\binom{i-l}{n-k}\Big/\binom{i}{n}$, if i ($\ge \alpha l$) is much larger than n, then it can be approximated by the binomial distribution [23] as

$$\Pr(C=k \mid B=n, A=i)\approx\binom{n}{k}\left(\frac{l}{i}\right)^k\left(1-\frac{l}{i}\right)^{n-k}. \tag{C5}$$

If we assume that the attempt to any known addresses is always successful, then we have

$$\begin{aligned}&\Pr(Y_1=z \mid C=k, B=n, A=i)\\ &\quad=\begin{cases}\binom{n-k}{z-k}p_1^{z-k}(1-p_1)^{n-z}, & \text{for } z\ge k,\\ 0, & \text{for } z<k,\end{cases}\end{aligned} \tag{C6}$$

where p_1 is the attempt success probability of a scanner for random IP addresses and $p_1 \le \theta_1$. Combining (C4), (C5), and (C6) yields

$$\Pr(Y_1=z \mid B=n, A=i)=\binom{n}{z}\left(\hat{p}_1(i)\right)^z\left(1-\hat{p}_1(i)\right)^{n-z}, \tag{C7}$$

where $\hat{p}_1(i)=p_1+(1-p_1)l/i$. Let $Y_1^*(n,i)$ denote $Y_1 \mid B=n, A=i$, then $Y_1^*(n,i) \sim$ Binomial(n, $\hat{p}_1(i)$) by (C7).

In order to evaluate (C3), we now calculate $\Pr(B = n \mid A = i)$. Since $\Pr(B = n \mid A = i) = \Pr(B = n, A = i) / \Pr(A = i)$ and $\Pr(A = i)$ is given by (C1), we need to evaluate $\Pr(B = n, A = i)$. Since B and A are reflecting sampled attempts and total attempts, respectively, until n distinct attempts are registered in the Bloom filter V_1, the event of $B = n$ and $A = i$ means all of the n sampled attempts are registered in V_1 without collision and all other i-n attempts are not sampled. In addition, the final attempt is always sampled and registered without collision by the definition of B and A. Thus, $\Pr(B = n, A = i)$ can be evaluated as

$$\Pr(B = n, A = i) = \binom{i-1}{n-1} (p_s(1-p_c))^n (1-p_s)^{i-n}. \tag{C8}$$

From (C1), (C8), and the definition of conditional probability, we have

$$\Pr(B = n \mid A = i) = \left(\frac{1-p_s}{1-p_s(1-p_c)} \right)^{i-n}. \tag{C9}$$

From (C3) and (C9), we can obtain

$$\Pr(\frac{Y_1}{n} > \eta \mid A = i) \le \Pr(\frac{Y_1}{n} > \eta \mid B = n, A = i) + 1 - \left(\frac{1-p_s}{1-p_s(1-p_c)} \right)^{i-n}. \tag{C10}$$

We need to select α larger than $(1-\theta_1)/(\theta_2-\theta_1)$ so that the increased success probability of the scanner $\hat{p}_1(i)$ $(= p_1 + (1-p_1)l/i)$ can not be larger than the minimum success probability of a normal host θ_2. Since $Y_1^*(n,i) \sim \text{Binomial}(n, \hat{p}_1(i))$ and $\hat{p}_1(i)$ decreases as i increases, by Lemma 1 we can obtain

$$\Pr(\frac{Y_1}{n} > \eta \mid B = n, A = i) \ge \Pr(\frac{Y_1}{n} > \eta \mid B = n, A = k), \quad \text{for } k \ge i. \tag{C11}$$

From (C2), (C10), and (C11), the following relation can be derived:

$$\Pr(\frac{Y_1}{n} > \eta) \le 1 - (1-p_c)^n + \sum_{i=n}^{\alpha l - 1} \binom{i-1}{n-1} p^n (1-p)^{i-n} + \Pr(\frac{Y_1}{n} > \eta \mid B = n, A = \alpha l). \tag{C12}$$

We assumed that α is selected such that αl is an integer. In the above relation, $Y_1 \mid_{B=n, A=\alpha l} = Y_1^*(n, \alpha l) \sim \text{Binomial}(n, p_1 + (1-p_1)/\alpha)$. If we $\hat{Y}_1(n, \alpha l)$ denote $Y_1^*(n, \alpha l)$ especially when p_1 has the maximum value of θ_1, then $\theta_1 + (1-\theta_1)/\alpha \ge p_1 + (1-p_1)/\alpha$ and by Lemma 1, we have

$$\Pr(\hat{Y}_1(n, \alpha l) > n\eta) \ge \Pr(Y_1^*(n, \alpha l) > n\eta). \tag{C13}$$

Combining (C12) and (C13) yields

$$\Pr(\frac{Y_1}{n} > \eta) \le 1 - (1-p_c)^n + \sum_{i=n}^{\alpha l - 1} \binom{i-1}{n-1} p^n (1-p)^{i-n} + \Pr(\hat{Y}_1(n, \alpha l) > n\eta). \tag{C14}$$

From (3) and (C14), we can know that if (21), (22), and (23) are satisfied, then the false negative probability constraint of (3) is satisfied.

The false negative probability is not affected by the known IP addresses of scanners since it is determined by the behavior of benign hosts. We can show the following relation is valid in the same way as that used for Y_2 in Appendix B although we need to condition one more time about the sampled attempts B between A and R:

$$\Pr(\frac{Y_2}{n} \le \eta) \le \Pr(Y_2'(n) \le n\eta). \tag{C15}$$

Thus, if (24) is satisfied, then the false positive probability is guaranteed by (C15). n