Connectionless Port Scan Detection on the Backbone

Avinash Sridharan*
University of Southern California
Dept. of Electrical Engineering
asridhar@usc.edu

Tao Ye
Sprint ATL
1 Adrian Ct.
Burlingame, CA, USA
tao.ye@sprint.com

Supratik Bhattacharyya
Sprint ATL
1 Adrian Ct.
Burlingame, CA, USA
supratik@sprint.com

ABSTRACT

Considerable research has been done on detecting and blocking portscan activities that are typically conducted by infected hosts to discover other vulnerable hosts. However, the focus has been on enterprise gateway-level Intrusion Detection Systems where the traffic volume is low and network configuration information is readily available. This paper investigates the effectiveness of existing portscan detection algorithms in the context of a large transit backbone network and proposes a new algorithm that meets the demands of aggregated high speed backbone traffic. Specifically, we evaluate two existing approaches - the portscan detection algorithm in SNORT [8], and a modified version of the TRW algorithm [6] that is a part of the intrusion detection tool BRO [12]. We then propose a new approach, TAPS, which uses sequential hypothesis testing to detect hosts that exhibit abnormal access patterns in terms of destination hosts and destination ports. We perform a comparative analysis of these three approaches using real backbone packet traces, and find that TAPS exhibits the best performance in terms of catching the maximum number of true scanners and yielding the least number of false positives. We have a working implementation of TAPS on our monitoring platform. Further implementation optimizations using bloom filters are identified and discussed.

1. INTRODUCTION

The spread of malicious, self-propagating code, popularly known as worms [1] [2] [3], continues to pose a grave threat to the security of the Internet. These worms exploit software vulnerabilities on certain TCP/UDP service ports to compromise end-hosts and install malicious code on them. Subsequently, the compromised hosts may be used to launch a wide-range of malicious activities such as denial of service attacks, sending spam emails, etc.

Stopping the propagation of these worms is a daunting task. Once a new software vulnerability is discovered, new worms exploiting the vulnerability appear in rapid succession and can spread throughout the Internet in a matter of hours. Furthermore, existing worms keep morphing to take on new forms and evade detection. And it is practically infeasible to ensure that the millions of hosts on the Internet are all properly configured and patched to make them secure against known vulnerabilities.

At the same time, a strategy of containment is necessary to limit the spread of worms at an early stage of propagation. Typically, the spread of a worm is preceded by a reconnaissance phase where infected hosts scan for new hosts that are vulnerable. This scanning activity (popularly known as port scanning) takes the form of probe packets targeted at specific TCP or UDP ports on target hosts. If the target host responds, then the scanning host may initiate the process of uploading malcode onto the target host.

In this paper, we examine approaches for detecting port scanning in transit backbone networks. Prior work on port scan detection algorithms has focused mainly on doing so at the point of entry into a stub network (e.g., an enterprise or a campus). There are a number of reasons for detecting port scans in transit networks rather than at the edge of individual stub networks. A transit network ISP could be interested in scanning activities originating from peering networks, therefore it could deploy detection tools on peering links. Moreover, a transit network carries a far more diverse traffic mix and may detect a wider range of scanning activities than an individual stub network. A transit network is also likely to observe a higher rate of scanning if the scans are targeted at hosts in several downstream stub networks. The study in [5] suggests that a global view of the traffic seen by transit networks could capture the patterns presented by scanners better. Finally, a stub network such as an enterprise may out source the detection task to its upstream transit provider due to lack of resources or expertise.

There are several important differences between detecting port scans in a transit network and in an enterprise-level stub network. Foremost, link speeds at the core of a transit network are much higher (e.g., 10 Gbps) than at the edge, where it connects to a stub network. Therefore, a detection algorithm has to be fast in order to scale to high-speed links. A transit network also sees a wider diversity of IP addresses, potentially generating a much larger amount of state that needs to be stored and accessed by a detection algorithm. Hence the algorithm has to be space-efficient. Third, unlike in a stub network, the detection algorithm cannot assume to have knowledge about the configuration of end-hosts and the topology of the network. Some algorithms (e.g., TRW [6]) that targets enterprise-level networks assume such knowledge. Fourth, the routing policies of a backbone often lead to asymmetric routing, which yields a unidirectional view of traffic at a given observation point. This makes it impossible to use detection algorithms that rely on statefull traffic analysis. Finally, a port scan detection algorithm in a transit network has to coexist with other monitoring and anomaly detection tasks, and must not interfere with these

*The author was at Sprint Advanced Technology Labs during the course of this work.

1-4244-0198-4/06/$20.00 ©2006 IEEE

tasks.

In our work, we start with two existing port scan detection algorithms. The first is the algorithm used by the popular intrusion detection system SNORT [8]. This algorithm is quite simple - mark a source IP address as a scanner if it contacts more than K distinct IP addresses within T seconds. While SNORT is widely used for enterprise-level networks, we evaluate its efficacy for the traffic mix typically found in a large transit network. The second algorithm we evaluate is a modified version of Threshold Random Walk(TRW) algorithm proposed in [6]. The modifications have been necessitated for its adaptation to a transit network, since the original algorithm assumes a bidirectional view of traffic and complete knowledge of the configuration of all end-hosts (e.g., in an enterprise-level network). Finally, we propose a new algorithm - Time-based Access Pattern Sequential hypothesis testing or TAPS. Briefly, this algorithm evaluates the access pattern of a source IP address during a given time interval in terms of the ratio of the number of distinct destination IP addresses and the number of destination ports accessed. It then applies a sequential hypothesis test on the access patterns over multiple time intervals to determine if a source IP is conducting scanning activities. We evaluate all three algorithms using packet-level traces collected from backbone links in a large transit network [4].

Results show that TAPS generates the maximum number of true scanners and minimum number of false scanners for both low and high traffic links. TAPS is much less sensitive to the rate limiting parameter changes than SNORT is. TAPS is also the best in catching low rate scanners.

The contributions of this paper are:

- We identify some important issues involved in performing port scan detection in a transit network, and propose modifications to the Threshold Random Walk Algorithm (TRW) to adapt it to a transit network.
- We propose a novel algorithm, TAPS, that detects port scans using access patterns of source IP addresses over successive time intervals and constructing a sequential hypothesis test over these patterns. We also present bounds on the scanning rates that this algorithm can detect.
- We conduct a comparative analysis of SNORT, TAPS and TRW to develop insights into how to detect port scans on high-speed transit network links.

The rest of the paper is organized as follows; Section 2 presents the related work in the development of port scan detection algorithms. Section 3 explains the testing environment setup. Section 4, Section 5, and Section 6 present SNORT, TRW, and TAPS algorithms, respectively. Section 7 performs a comparative evaluation between TAPS, TRWSYN and SNORT. Finally we conclude with section 8.

2. RELATED WORKS

TRW [6] is based on the observation that scanners who have little information about the characteristics of the network (which services are running on which IP's) are bound to have a higher number of failed connections than benign hosts that use the valid services in a network. SNORT, an open source software IDS, relies on simple rules such as N connections in M seconds to tag a source as a SCANNER. However, it is a very commonly used IDS software and the *de facto* standard in small stub networks. Thus we choose these two techniques for comparison. They will be discussed in detail in Section 4 and Section 5.

SPICE [9] and Leckie *et al* [10] propose probabilistic models to measure the likelihood of a source being a scanner. The probabilistic models try to assign a priori probabilities to IP's existing within the network, using normal traffic conditions for bench marking the probabilities. Scanners are detected by comparing the access probabilities of source IP's targeting this network with the a priori probabilities assigned to the host IP's. This technique is clearly not scalable, especially in case of the backbone, where the number of hosts is too large to maintain any information about them. Robertson *et al* [11] shares TRW's philosophy of doing statefull analysis of the underlying traffic. However, the algorithms are based on absolute thresholds rather than TRW's statistical thresholds, hence are less sophisticated and robust than TRW's detection algorithm.

BRO [12] is an open source deployed IDS system. TRW has been implemented on BRO [13]. This makes BRO more accurate than SNORT in terms of port scan detection at the edges. However, it adds a significant overhead as BRO must now maintain connection state and network topology information.

Honeypot mechanisms are used in some experimental capacity to catch port scanners. Honeypots are a set of IP addresses that are not allocated to any host, hence an access to this set of IPs indicates the rogue scanning activity from the source. Since a Honeypot generally consists of a cluster of machines acting as a decoy, it is expensive to deploy. There have been solutions proposed in [14] and [15] to reduce the cost of deployment of Honeypots. Honeypots also presents a myopic view of the overall traffic because scanners that don't access a Honeypot IP are not detected.

3. EXPERIMENT SETUP

We compare three detection algorithms: SNORT, TRW and TAPS. Of these, SNORT and TRW are existing algorithms and TAPS is proposed by us (see Section 6). We implement TRW and TAPS algorithms in the context of offline packet trace processing. The SNORT software is downloaded and adapted to run on our traces. Packet traces are collected by IPMON [4], the monitoring system deployed in the operational backbone of a tier one ISP SprintLink, is a passive monitoring system that captures the first 44 bytes of the IP header of every IP packet traversing a monitored link. This offline trace driven simulation environment allows us to compare all algorithms fairly with real traffic. TAPS has a working implementation on our monitoring platform as well.

For TRW and TAPS algorithms, we first perform flow grouping on all packets in the traces before gathering scanning statistics, using a regular five tuple (SRCIP, DSTIP, SRCPORT, DSTPORT, PROTO) flow definition. A flow terminates if we do not observe a packet belonging to this flow for a certain timeout period. (The timeout value defaults to 60s).

In particular, the packets are grouped into flows using the following rules:

- A packet is said to belong to a flow if it matches the

five tuple of the flow

- A flow is said to start at the time the first packet belonging to the flow was observed
- A flow is said to have finished if we do not observe a packet belonging to this flow for a certain timeout period. (The timeout value defaults to 60s)
- End time of a flow is the time at which the last packet in the flow was observed.

The following sections describe the three algorithms and their implementation details in the trace-driven simulation environment.

4. SNORT

SNORT [8] is a popular and widely deployed IDS system. The version of SNORT [8] we use is SNORTv2.2. Since IPMON collects only 44 bytes of the header, SNORT is modified slightly to accept these partial packets, i.e. malformed packet alert in the decoding engine is turned off. We make use of a new feature in version 2, the flow port scan preprocessor, to speed up the scan detection. The preprocessor counts the number of unique connections made by a source in order to classify it as a scanner. A unique connection for a given source is defined as a three tuple (DSTIP,DSTPORT, and PROTO). The relevant parameters and their values are described below; a further description of the functionality of each parameter can be seen in [8]:

- *server-watchnet*: This is the network that SNORT is monitoring. Since we do not want SNORT to save state of the whole network (which could involve a considerable amount of memory), we set the watchnet to 0.0.0.0 .
- *talker-fixed-window* t: Default value = 30 secs.
- *talker-fixed-threshold* N: Default value = 15. The static nature of this value is a weakness of SNORT. This N value is the only one varied in the evaluation to change the performance of SNORT.
- *talker-sliding-window*:Default value = 30 secs.
- *talker-sliding-threshold*: Default value = 30. We always set this to be the same as talker-fixed-threshold parameter.

5. TRW AND ITS BACKBONE ADAPTATION TRWSYN

The Threshold Random Walk [6] algorithm proposed by Jung *et al* is based on Sequential Hypothesis Testing. It is shown to require an extremely low number (~ 6) of observed events to make a decision, thus achieving the fast detection speed. We summarize the key points of this technique here and refer the details to [6].

Sequential Hypothesis Testing. The sequential hypothesis test is a series of updates on a likelihood variable, given a series of events in sequence, to determine which of two possible sets H_0 or H_1 the entity belongs to. Here H_0 is the set of benign hosts and H_1 is the set of scanners. An entity is a source IP address in question. An event is represented by an indicator random variable Y, such that for event i

$$Y_i = \begin{cases} 0 & \text{event } i \text{ is successful} \\ 1 & \text{event } i \text{ is not successful} \end{cases}$$

There are four probabilities associated with the random variable Y:

$$Pr[Y = 0|H_0] = \theta_0, \quad Pr[Y = 1|H_0] = 1 - \theta_0$$
$$Pr[Y = 0|H_1] = \theta_1, \quad Pr[Y = 1|H_1] = 1 - \theta_1$$

For the sequential hypothesis test to be successful, it is important that there be a clear demarcation in the statistical behavior of the two sets H_0 and H_1, e.g. on an average set H_0 will have many more successful events than set H_1. Whenever an event occurs, the sequential hypothesis testing updates the likelihood ratio defined as follows:

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

where Y_i can take the value 1 or 0 depending on the result on the event.

As the entity walks through a series of events it would cross one of two thresholds $\eta_0 < 1 < \eta_1$. If $\Lambda(Y) > \eta_1$ then the test declares the entity to belong to set H_1 and if $\Lambda(Y) < \eta_0$ the entity belongs to set H_0. The values of these two thresholds defines the accuracy of the test. In [6] Jung *el al* define two probabilities P_F (probability of false positives) and P_D (probability of detection) for port scan detection. The relation between the two thresholds and these two probabilities is

$$\eta_1 \leq \frac{P_D}{P_F} \ , \ \eta_0 \geq \frac{1 - P_D}{1 - P_F}$$

Intuitively we would like to select P_D as high as possible and keep P_F as low as possible. For the rest of this work we set $P_D = 0.99$ and $P_F = 0.01$.

TRWSYN. TRW has the indicator random variable set to the following:

$$Y_i = \begin{cases} 0 & \text{connection } i \text{ is successful} \\ 1 & \text{connection } i \text{ has failed} \end{cases}$$

The assumption is that benign hosts will make many more successful connection attempts than scanners. In an enterprise level network the connection status of an attempt could be judged either by domain knowledge of the enterprise network (which services are running on which machines) or by maintaining connection status of the bidirectional traffic. On the backbone we observe aggregate traffic and therefore do not have domain specific information. Moreover, our view of the traffic is from unidirectional links. The state of a connection can thus be monitored only if the protocol itself is connection oriented, i.e. TCP but not UDP. We modify the 'connection' definition to accommodate these constraints using the fact that TCP's first packet should always have a SYN flag set. We call our adaptation TRWSYN.

In TRWSYN, single packet flows with the packet's SYN flag set are defined as failed connections; flows having more than one packet and single packet flows without a SYN flag are defined as successful connections for a source IP. Note that some 'successful connections' by this definition, for example a single packet flow without a SYN flag, might be

failed connections. However, without an oracle this is an inevitable inaccuracy TRWSYN suffers due to the constraints of the backbone monitoring environment.

In order to present the implementation of the TRWSYN algorithm, we elaborate on the update mechanism that forms the core of the hypothesis testing algorithm:

$$
\begin{aligned}
&if\ (flow.pkts > 1 \\
&\quad or\ (flow.pkts == 1\ and\ flow.flag! = SYN)) \\
&\quad \Lambda Y(flow.srcip)_i = \Lambda Y(flow.srcip)_{i-1} * \tfrac{Pr[Y_i=0|H_1]}{Pr[Y_i=0|H_0]} \\
&else\ if\ (flow.pkts == 1\ and\ flow.flag == SYN) \\
&\quad \Lambda Y(flow.srcip)_i = \Lambda Y(flow.srcip)_{i-1} * \tfrac{Pr[Y_i=1|H_1]}{Pr[Y_i=1|H_0]} \\
&if\ (\Lambda Y(flow.srcip) > \eta_1) \\
&\quad Add\ flow.srcip\ to\ the\ scanner\ list \\
&if\ (\Lambda Y(flow.srcip) < \eta_0) \\
&\quad Delete\ \Lambda Y(flow.srcip)\ from\ the\ cache
\end{aligned}
$$

The update mechanism is called every time a flow ends. Flows with single packet and SYN flag set are termed as failed connections and hence the likelihood ratio for a particular src IP $\Lambda Y(flow.srcip)$ is updated with an unsuccessful event ($Y = 1$). For all other flows the likelihood ratio is updated with a successful event ($Y = 0$). For a particular src IP when the likelihood ratio crosses the threshold η_1 it is added to the list of scanners and when the likelihood ratio crosses the threshold η_0 it is removed from the list of source IP's on which the sequential hypothesis test is being performed.

6. TAPS: TIME BASED ACCESS PATTERN SEQUENTIAL HYPOTHESIS TESTING

We design the TAPS (Time Based Access pattern Sequential Hypothesis Testing) port scan detection algorithm by combining the powerful Sequential Hypothesis Testing technique and a more general access pattern that is domain knowledge and protocol agnostic.

6.1 Access Patterns

We need to find a less constrained access pattern as the metric to differentiate between the behavior of malicious scanners and benign hosts for the TAPS algorithm. We observe that scanners go to many more IP's vs ports or vice versa as compared to benign hosts, i.e. a scanner would have $\frac{IP}{PORT} >> 1$ or $\frac{PORT}{IP} >> 1$. Note that we refer these fractions to destination IP and PORT from here on implicitly. Figure 1 shows the access patterns of a set of hosts for a backbone trace. We used known port signatures to find the full set of scanners. The access pattern of these scanners lie to the extreme top left corner (Figure 1, "HOST SCAN REGION") or extreme bottom right (Figure 1, "PORT SCAN REGION") corner of the graphs. 'Host Scan Region' illustrates the pattern of a scanner going to the same few ports on many different hosts, i.e. IP addresses. 'Port Scan Region' shows the pattern of a scanner going to many ports of a few hosts. Note the particular use of 'port scan' in this graph is more narrow than the general usage in this paper.

6.2 TAPS: Time-based Access Pattern Sequential Hypothesis Testing

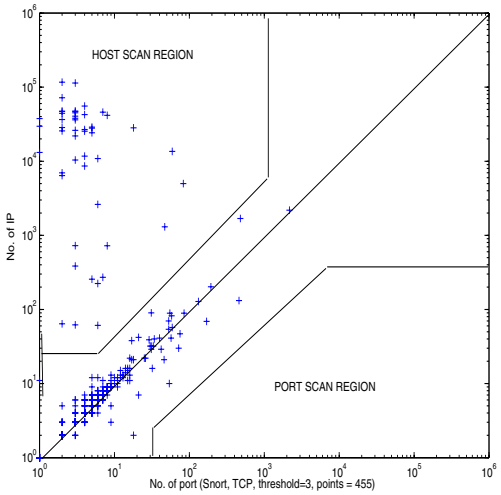


Figure 1: Access pattern of hosts on the backbone.

In order to make TAPS protocol independent we use the sequential hypothesis technique, with the above access pattern as the indicator random variable.

As per the metric of access pattern for a scanner:

$$IP/port\ >>\ 1\ or\ port/IP\ >>\ 1$$

The indicator random variable is defined as follows:

$$
Y_i = \begin{cases} 0 & if\ IP/port\ <=\ k \\ & and\ port/IP\ <=\ k(unsuccessful\ event) \\ 1 & if\ IP/port\ >\ k \\ & or\ port/IP\ >\ k(successful\ event) \end{cases}
$$

Recall from section 5, the likelihood ratio for the Sequential hypothesis testing is updated whenever an event occurs. We use time bins as the event generation mechanism, i.e. every N seconds the IP/port ratio is calculated as an event for each suspicious source. Thus the length of the time bins would play a crucial role in defining the results of the sequential hypothesis test. The choice of k is important for the false positive rate. We highlight these again during the evaluation of the TAPS algorithm.

The motivation for choosing time bins as the event is based on the observation that attackers need to maintain a certain scanning rate, e.g. number of hosts scanned per minute(/second), in order to achieve a certain propagation rate. By determining the size of the time bins, there is a possibility of giving guarantees on lower bounds for rates that can be detected. In section 6.3 we provide closed form expressions for determining lower bounds on scanning rates that can be detected by TAPS.

To further corroborate our intuition that scanners depict the behavior signified by hypothesis H_1, we analyze a set of scanners that we believe is the ground truth for our set of traces, identified by methods in Section 7.

Figure 3 presents the CDF of the percentage of time bins for which the $\frac{IP}{Port}$ ratio is greater than 2. This behavior of scanners was observed over a ten thousand second trace on an OC-48 link. The knee for the CDF plot lies around the 90% mark, which clearly implies that the hypothesis H_1 captures the behaviors of scanners.

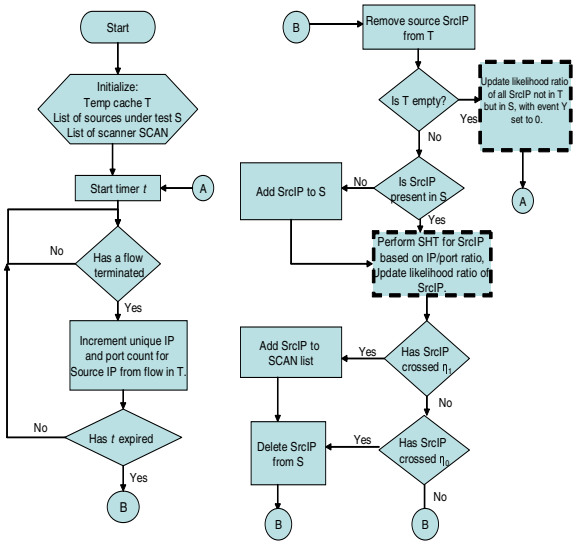


Figure 2: Algorithm used to implement TAPS.

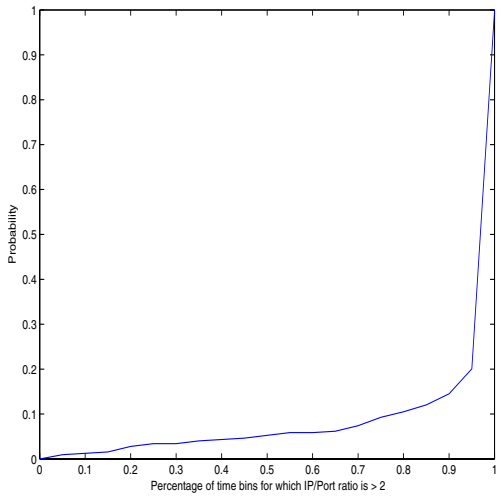


Figure 3: CDF of the percentage of time bins for known scanners for which $\frac{IP}{Port}$ ratio is > 2.

The implementation of TAPS is presented in Figure 2. Similar to the TRWSYN case, we have marked the update mechanism by a dashed bounding box and elaborate on the update mechanism for the likelihood ratio below:

$$if\ (flow.destipToPortRatio > k\ ||\ flow.destportToIpRatio > k)$$
$$\Lambda Y(flow.srcip)_i = \Lambda Y(flow.srcip)_{i-1} * \frac{Pr[Y_i=1|H_1]}{Pr[Y_i=1|H_0]}$$

$$else\ if\ (flow.destipToPortRatio <= k\ \&\&\ flow.destportToIpRatio <= k)$$
$$\Lambda Y(flow.srcip)_i = \Lambda Y(flow.srcip)_{i-1} * \frac{Pr[Y_i=0|H_1]}{Pr[Y_i=0|H_0]}$$

$$if\ \Lambda Y(flow.srcip) > \eta_1$$
$$Add\ flow.srcip\ to\ the\ scanner\ list$$

$$if\ \Lambda Y(flow.srcip) < \eta_0$$
$$Delete\ \Lambda Y(flow.srcip)\ from\ the\ cache$$

where

$$Pr[Y_i = 0|H_0] = \theta_0, \quad Pr[Y_i = 1|H_0] = 1 - \theta_0$$
$$Pr[Y_i = 0|H_1] = \theta_1, \quad Pr[Y_i = 1|H_1] = 1 - \theta_1$$

As per Figure 2 the update mechanism for the likelihood ratio would be called every time bin. For the duration of the time bin a hash table T is maintained with all the sources that have been observed with single packet flows, along with the IP's and ports that have been accessed. The likelihood ratio of each source IP $\Lambda Y(flow.srcip)$ is maintained in a hash table S. In the first conditional loop of the update mechanism, the $\frac{IP}{PORT}$ ratio and $\frac{PORT}{IP}$ are observed. If the condition holds then the likelihood ratio is updated with a successful event, otherwise the likelihood ratio is updated with an unsuccessful event. After updating the likelihood ratio, it is compared with the two thresholds η_1 or η_0. These two thresholds are defined at the end of Section 5. If the likelihood ratio crosses the threshold η_1 then it is added to the list of scanners, else it is removed from the hash table S, since it is considered to be benign.

6.3 Lower bounds on the expected scanning rate

We define the expected scanning rate of a scanner by:

$$scan\ rate = \frac{E[IP|H_1]}{E[N|H_1]} \tag{2}$$

$E[N|H_1]$ is the expected number of time bins before TAPS makes a decision that a source is a scanner. $E[IP|H_1]$ is the expected number of IP's that a scanner visits before it is tagged as a scanner.

Under the assumption that $\theta_1 = 1 - \theta_0$, we can model the hypothesis test as an n state random walk. By accounting for the minimum number of IP's that need to be accessed at each transition of the walk, we find the lower bound of the expected scanning rate that can be detected is:

$$\frac{E[IP|H_1]}{E[N|H_1]} = \frac{1}{E[N|H_1]} \left(\frac{k + \frac{\theta_1}{\theta_0}}{1 - \frac{\theta_0}{\theta_1}} \right) \left(\frac{\ln(\eta_1)}{\ln(\frac{1-\theta_1}{1-\theta_0})} - 1 \right) \tag{3}$$

The proof for equation (3) is presented in Appendix A. The proof and derivation for $E[N|H_1]$ is presented in [6].

We believe the minimum expected scanning rate is an important tuning parameter in TAPS which provides a tuning knob to the user for deciding scans of interest. We are able

to verify the correctness of this expression with the threshold (expected scanning rate) obtained by analyzing scanners missed by TAPS on an OC-48 link in section 7.4.

7. EVALUATION

In this section we evaluate the performance of SNORT, TRWSYN and TAPS, in terms of accuracy, completeness, parameter sensitivity and detection and miss characteristics.

A pre-requisite for evaluating the performance of any scan detection algorithm is the existence of a "ground truth" set of IP addresses that are true scanners. Since our traces contain only TCP/IP headers and not complete packet payloads, it is virtually impossible to guarantee that any algorithm can identify a source as a scanner with absolute certainty. We therefore resort to a "best effort" approach to define the scanning behavior through bootstrapping. Due to the protocol specific nature of TRWSYN, it is only suitable for detecting TCP scanners. Therefore we only generate the TCP scanner ground truth set for the accuracy comparison. We first run all three algorithms (SNORT, TRWSYN and TAPS) on the traces with very loose parameters to generate a superset of scanner candidates. From this set, we cross-check known scanning signatures(port 445, 135, etc), wide spread destination ports on one destination IP, and flow size against the intuitive notion of a scanner behavior. The sources corresponding to this behavior constitute our ground truth scanners. We use this ground truth set for each trace to identify success detection, false negatives, and false positives of an algorithm. Since the goal is to compare the performance of each algorithm, this approximation bootstrapping method should suffice in producing an unbiased ground truth for a fair comparison.

We only present representative results from two traces in the backbone. TRACE-1 is from a low volume OC-3 link that is close to an enterprise gateway; TRACE-2 is from a high volume OC-48 link. We include a low volume link in the study to highlight the difference in the algorithms' performance under high volume traffic and low volume traffic. For the duration of the trace, we confirmed that there was no routing changes.

7.1 Accuracy comparison

For a comparative analysis of TAPS and TRWSYN, the values chosen are; $\eta_1 = 99, \eta_0 = 0.01, P[Y = 0|H_0] = 0.8, P[Y = 0|H_1] = 0.2, P[Y = 1|H_0] = 0.2, P[Y = 1|H_1] = 0.8$. For TAPS k/t is set to 3/10sec for both traces. The parameter N for SNORT is selected to have the best result for each trace. Note from now on, unless explicitly stated (such as in section 7.2), the results only concern TCP scanners.

To arrive at a value of k for TAPS we analyzed the CCDF plot for the set of non-scanners that were obtained by subtracting the true scanners in the trace, presented in Figure 4. As is evident, the cut of value for the CCDF value lies around $IP/Port = 2$, i.e., it is highly unlikely that non-scanners will achieve a ratio of $\frac{IP}{Port} > 2$. Taking a slightly conservative estimate we set this value to 3.

Table 1 presents the comparative accuracy of the three algorithms. The metrics used to measure the accuracy are ***Success, False Positives*** and ***False Negatives***. ***Success*** is the number of detected scanners, ***False Positives*** are the number of falsely identified scanners, and ***False Negatives*** are the number of true scanners that were missed. Their corresponding ratio metrics compared to the true number of

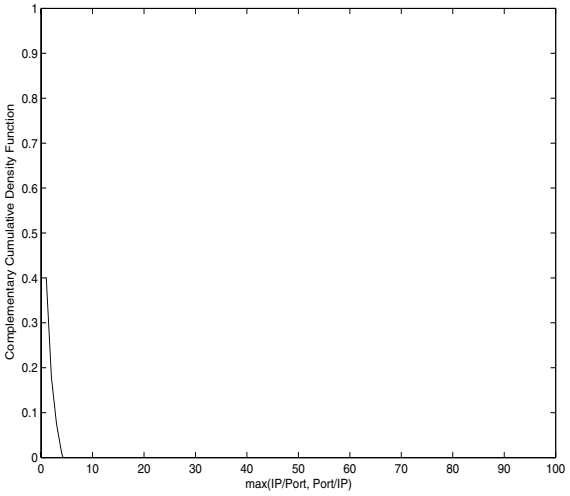


Figure 4: CCDF plot for the $\frac{IP}{Port}$ ratios for non-scanners.

scanners in the trace are defined as follows:

$$\text{Success Ratio } R_s = \frac{Number\ of\ true\ scanners\ detected}{Number\ of\ true\ scanners}$$
$$\text{False Positive Ratio } R_{f+} = \frac{Number\ of\ falsely\ ided\ scanners}{Number\ of\ true\ scanners}$$
$$\text{False Negative Ratio } R_{f-} = \frac{Number\ of\ true\ scanners\ missed}{Number\ of\ true\ scanners}$$

Success Ratio indicates the effectiveness of a detection algorithm, while False Negative Ratio indicates its completeness. We do not use *detection efficiency* defined as $\frac{Number\ of\ true\ scanners}{Number\ of total\ detections}$ in order to maintain the same denominator for all three ratios.

Overall, Table 1 shows that TAPS performed the best out of the three algorithms, with the lowest false positive and false negative ratio, and the highest success ratio of above 90%. It also shows that in the low traffic environment such as TRACE-1 (1Mbps), TRWSYN and TAPS both performed fairly well, with a success ratio above 80%, and false positive ratio below 25%. SNORT did not perform that well, giving a very high false positive ratio of 634.8%. However in the high traffic environment of TRACE-2 (100Mbps), SNORT's performance improved while that of TRWSYN and TAPS degraded. Between TRWSYN and TAPS, the performance of TRWSYN fell much more, doubling its false positive ratio and increasing its false negative ratio by 50%.

Table 1 shows the not so surprising result that SNORT has the highest of false positive and false negative ratio. The error made by SNORT is due to a mismatch of the scanner's scanning rate and SNORT's threshold scanning rate N/t. This rate is known to be difficult to set. If it is set too low, SNORT generates too many false positives; if it is set too high, SNORT generates too many false negatives. However, since TAPS is based on Sequential Hypothesis testing, which is a statistical technique, it is more robust to such parameter variations.

TRWSYN did not perform very well in our testing, mainly due the limited ability of statefull analysis of traffic on the backbone. The number of false positives given by TRWSYN is quite high. By hand examination, we find that many legitimate connections are categorized to be failed connections simply because they fall into our definition of 'one packet

TRACE-1(OC-3)		SNORT(N/t=3/30)	TRWSYN	TAPS(k/t=3/10)
	Success (R_s)	42 / 63.6%	55 / 83.3%	65 / 98.5%
	False Positive (R_f-)	419 / 634.8%	16 / 24.2%	6 /9.1%
	False Negative (R_f+)	24 / 36.3%	11 / 16.7%	1 / 1.5%
	True Scanners	66	66	66
TRACE-2(OC-48)		SNORT(N/t=60/30)	TRWSYN	TAPS(k/t=3/10)
	Success (R_s)	290 / 64.9%	332 / 74.3%	403 / 90.2%
	False Positive (R_f+)	365 / 81.7%	204 / 45.6%	44 / 9.8%
	False Negative (R_f-)	157 / 35.1%	115 / 25.7%	54 / 12.1%
	True Scanners	447	447	447

Table 1: Accuracy Comparison of Port Scan Detection Algorithms

TRACE-1 (No. of Scanners)	TRACE-2 (No. of Scanners)	Scanned Port
3	231	137
0	78	135
0	51	0
0	8	1434
4	77	vertical scanners

Table 2: UDP port scanners captured by TAPS for TRACE-1 and TRACE-2

flow with the SYN flag set'. This definition is proven to be too general. However, since it is effectively impossible to have an 'oracle' such as the one found in stub networks, there currently is no better definition of failed connection in the backbone. This weakness of TRW further confirms our motivation to design TAPS – a protocol and oracle independent port scan detection algorithm.

7.2 Completeness: Advantage of connectionless detection

Since TAPS is protocol independent, it has a considerable advantage over TRWSYN for detecting UDP scans. In fact some of the fastest spreading worms, e.g. Slammer and Blaster, spread through UDP scans and could not have been detected by TRWSYN. In Table 2 we show the UDP scanners that were detected using TAPS. To keep false positives at a minimum, we used a stronger set of parameters (k/t = 10/10) than in the TCP scanner detection shown in Table 1. We further confirm that most of these scanners demonstrate a pattern of sequentially scanning destination IP addresses in a sub-net, i.e. x.x.x.1, x.x.x.2, x.x.x.3, etc. Others perform vertical scans, which means a large number of ports of the same host are stepped through in a very short time. In TRACE-1, TAPS detected an additional 7 UDP scanners. 3 of these display a strong scanning pattern, scanning port 137 of a sub-net's addresses sequentially. The other 4 are strong vertical scanners. In TRACE-2 we observed a total of 445 UDP scanners, of which 231 are port 137 scanners and 77 vertical scanners. These results show that TRWSYN missed obvious scanners simply because the scanners use a connection less protocol UDP. However, the protocol independent TAPS is capable of catching them.

7.3 Rate limiting sensitivity

Both TAPS and SNORT are essentially rate limiting detection methods. Figure 5 shows the detection accuracy of TAPS and SNORT under a series of rate limiting factor parameter variations. We define 'rate limiting factor' to be k/t ($\#IP/\#port$ over t seconds, see section 6) for TAPS and N/t (#IP over t seconds, see section 4) for SNORT. TAPS' false positive rate and false negative rate are shown to be much less sensitive to the change in k/t, varying between 10% to 30%. SNORT however changes much more dramatically with the k/t, between 30% to 300%.

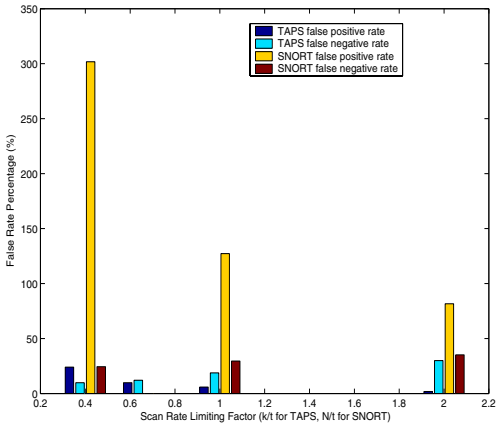


Figure 5: Rate Limiting factor sensitivity comparison between SNORT and TAPS

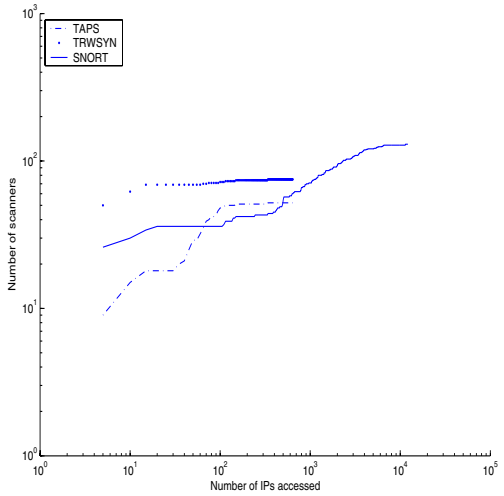
Traditionally the strength of rate limiting methods is that they require no connection information, and hence are easy to implement. However the weakness is that parameter setting is static and causes huge inaccuracies. In addition to a more insightful access pattern, TAPS takes advantage of the statistical method Sequential Hypothesis Testing to overcome the weakness of SNORT-like rate limiting methods. It is much easier to set the parameter for correct detection of scanners. In fact k is the same for TAPS for both the low traffic TRACE-1 and high traffic TRACE-2. However in SNORT N had to be set to 3 for TRACE-1 and 60 for TRACE-2 to achieve a good detection rate and false positive rate balance.

7.4 Detection and Miss characteristics

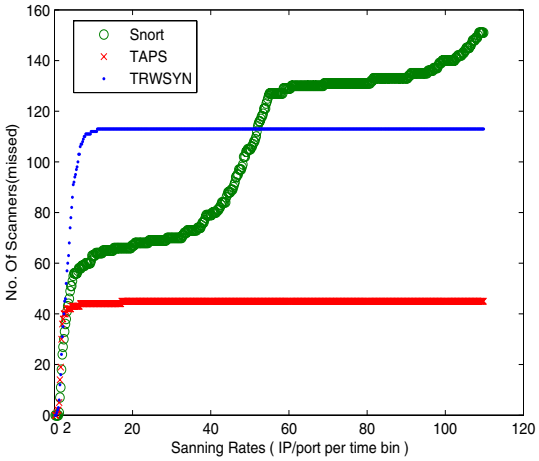
While the simple accuracy comparison in section 7.1 shows the relative effectiveness of different algorithms, it does not explain the results. We take a closer look at the characteristics of the scanners each algorithm detects and misses. There are many behavior statistics one could study, such as aggressiveness, persistence, etc. We isolate two differenciating factors – target footprint and scanning rate to illustrate below. A scanner's target footprint is a measurement of how many IP addresses it tries to reach. The scanning rate shows how fast a scanner can reach these addresses.

Figure 6(a) plots the cumulative distribution function of the number of IP addresses reached by all the missed scanners (false negatives) for each algorithm. First, we see that the TAPS line is lower than TRWSYN and SNORT, showing that it missed the least number of scanners. More specifi-

cally, it missed the least number of low footprint (reaching fewer than 20 IPs) scanners. Second, figure 6(a) shows that SNORT is the longest in x-axis, showing it could miss very aggressive scanners reaching large target footprints. This is possible because scanners with rates just lower than the SNORT threshold could in fact reach a large footprint over a long period of time. TRWSYN missed the highest number of low (fewer than 20 IPs) and medium(fewer than 100 IPs) footprint scanners. This is again due to the imperfect definition of 'failed connection' on the backbone. This is a clear drawback for TRWSYN on the backbone.



(a) Target footprint of scanners missed(false negative) by TAPS, SNORT and TRWSYN.



(b) Distribution of scanning rate for missed scanners for TAPS, TRWSYN and SNORT

To further analyze the miss characteristics we plot the distribution of the scanning rate for the missed scanners in Figure 6(b) for each of the three algorithms. Here we use a notation similar to 'rate limiting factor' from the previous section as scanning rate S. S is defined as $\frac{IP}{port}$ per time bin. The reasoning for such a definition, following the ideology of TAPS, is that in addition to scanning ports, suspicious hosts could show normal network connection behavior as well. We believe this metric filters out the normal behavior and captures the anomalous behavior. Figure 6(b) shows again that SNORT misses the maximum number of scanners with a high scanning rate, as the knee of the curve is around $S = 100$. TRWSYN and TAPS have similar limit on scanning rates, with TRWSYN slightly higher at $S = 7$ and TAPS at $S = 2$. Moreover, TRWSYN missed more scanners with the same scanning rate than TAPS.

In equation (3) we show the analytical value of the minimum expected scanning rate that TAPS can detect. By setting $k = 3, \theta_1 = 0.2, \theta_0 = 0.8, \eta_1 = 99, \eta_0 = 0.01$ and $E[N|H_1] = 6$ (average number of time bins required to make a decision as given in [6] for the specified parameters) in equation (3) we get $\frac{IP\ to\ port\ ratio}{time\ bins} \approx 1.7$. This matches well with the knee of the curve for TAPS in Figure 6(b) . Hence the scanners missed had a scanning rate lower than the bound that TAPS was designed to detect. This observation highlights the importance of this design parameter for TAPS and the flexibility that it provides to the user.

8. CONCLUSION

A significant body of work exists on algorithms for port scan detection. However, very little attention has been paid to designing algorithms tailored for a transit network. In this paper we address this deficiency by evaluating two popular port scan detection algorithms - SNORT and a modified version of TRW, and proposing a new algorithm, TAPS. Unlike TRW, TAPs does not assume any knowledge about the configuration of the network or require a bi-directional view of traffic, and therefore performs better than TRW for high-speed backbone links. It uses sequential hypothesis testing to reduce the impact of parameter value choices. This is unlike SNORT, where the performance is much more dependent on tailoring the parameter values on the data set in question. Hence our work provides insights into the issues surrounding the design of a port scan detection algorithm for high-speed transit networks.

There are several directions in which this work can be extended. We are in the process of validating our findings on additional backbone packet traces. The construction of a "ground truth" set of scanners is a challenging problem. Currently a large amount of additional information needs to be collected and processed to create a comprehensive set of true scanners. Finally, we need to study the scanners identified by each algorithm in greater depth with respect to characteristics such as their aggressiveness, persistence, etc. This will shed more insights into the scanners caught or missed by each algorithm, and can potentially lead to the development of more sophisticated algorithms in the future.

9. REFERENCES

[1] http://securityresponse.symantec.com/ /avcenter/venc/data/w32.blaster.worm.html
[2] http://www.cert.org/advisories/CA-2001-19.html
[3] http://securityresponse.symantec.com /avcenter/venc/data/w32.sasser.worm.html
[4] Chuck Fraleigh, Sue Moon, Brian Lyles, Chase Cotton, Mujahid Khan, Rob Rockell, Deborah Moll, Ted Seely and Christophe Diot. Packet-Level Traffic

Measurements from the Sprint IP Backbone". In *IEEE Network Magazine, 2003*

[5] Vinod Yegneswaran, Paul Barford and Johannes Ullrich. Internet Intrusions: Global Characteristics and Prevalence. In *Proceedings of ACM SIGMETRICS 2003*

[6] Jaeyeon Jung, Vern Paxson, Arthur W. Berger, and Hari Balakrishnan: Fast Portscan Detection Using Sequential Hypothesis Testing. In *Proc. IEEE Symposium on Security and Privacy, 2004*

[7] C. Fraleigh and S. Moon and B. Lyles and C. Cotton and M. Khan and D. Moll and R. Rockell and T. Seely and C. Diot. Packet-Level Traffic Measurements from the Sprint IP Backbone. *IEEE Network Magazine, vol. 17, no. 6, Nov. 2003.*

[8] http://www.snort.org

[9] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. In *Proceedings of the 7th ACM Conference on Comuter and Communications Security, Athens, Greece, 2000*

[10] C. Leckie and R. Kotagiri. A probabilisitc approach to detecting network scans. In *Proceedings of the Eighth IEEE Network Operations and Management Symposium (NOMS 2002), Pages 359-372, Florence, Italy, Apr. 2002.*

[11] S. Robertson, E. V. Siegel, M. Miller, and S. J. Stolfo. Surveillance detection in high bandwidth environments. In *Proceedings of the 2003 DARPA DISCEX III Conference, pages 130-139, Washington, DC, 2003. IEEE Press. 22-24 April 2003.*

[12] http://www.icir.org/vern/bro-info.html

[13] Nicholas Weaver, International Computer Science Institute; Stuart Staniford, Nevis Networks; Vern Paxson, International Computer Science Institute and Lawrence Berkeley National Laboratory. Very Fast Containment of Scanning Worms. In *Proceeding of 13th Usenix Security Symposium 2004*

[14] A Virtual Honeypot Framework. In *Proceedings of the 13th USENIX Security Symposium*

[15] Xuxian Jiang and Dongyan Xu. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *Proceedings of the 13th USENIX Security Symposium*

[16] Jainning Mai, Ashwin Sridharan, Chen-Nee Chuah, Tao Ye and Hui Zang.(Sprint Nextel Advanced Technology Labs. On the Impact of Packet Sampling on Anomaly Detection. *Technical Report RR05-ATL-070679, 2005*

[17] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. Internet Mathematics Vol. I, No. 4: 485-509

[18] http://www.linklogger.com/TCP1025.htm

APPENDIX

A. BOUNDS ON EXPECTED DETECTABLE SCANNING RATES

Note that the sequential hypothesis is a random walk. We start the test by setting the likelihood ratio ($\Lambda(Y)$) to 1. Every time bin if the scanner accesses greater than k IP's then we move 1 step up by $\frac{1-\theta_1}{1-\theta_0}$. If the scanner accesses less than k IP's then we move one step down by $\frac{\theta_1}{\theta_0}$. The movement of the likelihood ratio above or below is probabilistic and not deterministic and hence the test is a random walk on the values of the likelihood ratio. This random walk can be modeled as a discrete time Markov chain. Each state in the Markov chain maps to a value of the likelihood ratio. By equation (1)

$$\Lambda(Y) = \prod \frac{Pr(Y_i|H_1)}{Pr(Y_i|H_0)}$$

Hence

$$\ln(\Lambda(Y)) = \sum \ln \frac{Pr(Y_i|H_1)}{Pr(Y_i|H_0)}$$

where

$$\frac{Pr[Y=1|H_1]}{Pr[Y=1|H_0]} = \frac{1-\theta_1}{1-\theta_0}$$

and

$$\frac{Pr[Y=0|H_1]}{Pr[Y=0|H_0]} = \frac{\theta_1}{\theta_0}$$

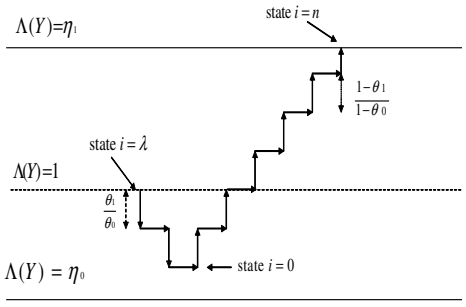


Figure 6: An example of a walk that a scanner might take to go from state λ to state n.

Given that we want to model the walk of a scanner that we detect, the minimum value of the likelihood ratio that the scanner achieves is $\left(\eta_0 + \frac{\theta_1}{\theta_0}\right)$ (if the likelihood ratio had achieved η_0 it would have been tagged benign). Similarly the maximum value of the likelihood ratio that the scanner can achieve is η_1 (at this point the source IP has been tagged as a scanner). Figure 6 gives a visual representation of a possible path that a scanner might take to go from state λ to state n.

Under the assumption that $\theta_1 = 1-\theta_0$, we can define this walk as an n state walk where 0 is the lowest state possible and n is the highest state possible. This assumption holds true especially in the case of TAPS where the behavior of scanners and non-scanners complement each other for the metric chosen (access patterns). We also define the state λ, such that $0 < \lambda < n$ to be the state from which the walk

begins ($\Lambda(Y) = 1$). Further we can map each state of the walk to the logarithm of a particular value of the likelihood ratio($\Lambda(Y)$). Thus the relationship between the logarithm of the likelihood ratio and a state i of the walk can be given as follows:

$$\ln(\Lambda(Y_i)) = \begin{cases} 0 & i = \lambda \\ \ln(\frac{1-\theta_1}{1-\theta_0}) \times (i-\lambda) & \lambda < i < n \\ \ln(\frac{\theta_1}{\theta_0}) \times (\lambda - i) & 0 < i < \lambda \end{cases}$$

n the maximum number of states for this Markov chain would be given by:

$$n = \frac{\ln(\eta_1)}{\ln(\frac{1-\theta_1}{1-\theta_0})} + \frac{\ln(\eta_0 + \frac{\theta_1}{\theta_0})}{\ln(\frac{\theta_1}{\theta_0})}$$

The above equation can be understood by keeping in mind the fact that when the TAPS algorithm starts for each source IP i, $\Lambda Y(srcip_i)$ is initialized to 1. Hence within a finite number of steps $\frac{\ln(\eta_1)}{\ln(\frac{1-\theta_1}{1-\theta_0})}$ you can cross the threshold η_1. Similarly by proceeding a finite number of steps $\frac{\ln(\eta_0+\frac{\theta_1}{\theta_0})}{\ln(\frac{\theta_1}{\theta_0})}$ we will be one step away from the threshold η_0. Thus the number of steps required to go from state λ to state n is:

$$n - \lambda = \frac{\ln(\eta_1)}{\ln(\frac{1-\theta_1}{1-\theta_0})} \tag{4}$$

and the number of steps to go from state 0 to state λ would be

$$\lambda = \frac{\ln(\eta_0 + \frac{\theta_1}{\theta_0})}{\ln(\frac{\theta_1}{\theta_0})} \tag{5}$$

We want to calculate the expected scanning rate that can be detected by TAPS. The expected scanning rate can be given as:

$$scan\ rate = \frac{E[IP|H_1]}{E[N|H_1]} \tag{6}$$

Where $E[IP|H_1]$ is the expected number of destination IP's that a source IP has gone to given that it is a scanner and $E[N|H_1]$ is the expected number of time bins required to come to a decision that the source is a scanner. We first calculate $E[IP|H_1]$. An important aspect of the sequential hypothesis that we will be using in order to calculate $E[IP|H_1]$ is that to go from state i to $i+1$, the source IP needs to have accessed at least k destination IP's. Similarly to go from state i to $i-1$ the source IP needs to have accessed at least 1 IP. Note that by accounting for the minimum number of IP's accessed during a state transition we are calculating a lower bound on the expected scanning rate that can be detected by TAPS.

The transition probabilities of the Markov chain will be given by:

$$P(1,0) = 0, \quad P_{i,i+1} = p, \quad P_{i,i-1} = 1 - p = q, \quad 1 \le i < n$$

where

$$p = Pr[Y = 1|H_1] = 1 - \theta_1$$

We set $P(1,0) = 0$ since we want to calculate the expectation $E[IP|H_1]$ under the assumption that the scanner is detected. This can happen only when the scanner does not reach state 0.

Let J_i be a random variable that denotes the number of IP when going from state i to $i+1$. Since the random walk is modeled as a Markov chain the variable $J_i, i = 0,\ 1,\ 2,\ \ldots, n-1$ are independent. If $J_{\lambda,n}$(we start our walk from state λ) is the number of IP's that are accessed by a source IP when going from state λ to state n, then

$$Q_{\lambda,n} = \sum_{i=\lambda}^{n-1} J_i \tag{7}$$

Letting $\gamma_i = E[J_i]$ and conditioning on the transition of the next state we get

$$\gamma_i = kp\ +\ E[1\ +\ J^*_{i-1}\ +\ J^*_i]q \tag{8}$$

Where J^*_{i-1} and J^*_i are the of IP's accessed during the transition from state $i-1$ to i and then from state i to $i+1$. By the Markovian property $E[J^*_i] = E[J_i]$ and $E[J^*_{i-1}] = E[J_{i-1}]$. Hence

$$\begin{aligned} \gamma_i &= kp\ +\ q\ +\ q(\gamma_{i-1} + \gamma_i) \\ &= k\ +\ \tfrac{q}{p}(1 + \gamma_{i-1}) \end{aligned} \tag{9}$$

Setting $\frac{q}{p} = \alpha$ we get

$$\gamma_i = \ k\ +\ \alpha(1 + \gamma_{i-1})$$

Since $\gamma_0 = k$ we get

$$\begin{aligned} \gamma_1 &= k + (1+k)\alpha \\ \gamma_2 &= k + (1+k)\alpha(1+\alpha) \\ &\vdots \quad \vdots \\ \gamma_i &= k + \alpha(k+1)\textstyle\sum_{j=0}^{i-1} \alpha^{j-1} \end{aligned}$$

Hence by (7)

$$\begin{aligned} E[Q_{\lambda,n}] &= \textstyle\sum_{i=\lambda}^{n-1}[k + \alpha(k+1)\frac{(1-\alpha^i)}{(1-\alpha)}] \\ &= k(n-\lambda-1)\ + \\ &\alpha\tfrac{(k+1)}{(1-\alpha)}(n-\lambda-1) - \tfrac{(k+1)\alpha}{(1-\alpha)}\textstyle\sum_{i=\lambda}^{n-1}\alpha^i \\ &= k(n-\lambda-1) \\ &+\tfrac{(k+1)\alpha}{(1-\alpha)}(n-\lambda-1) - \tfrac{(k+1)}{(1-\alpha)^2}(\alpha^{\lambda+1} - \alpha^{n+1}) \\ &\approx \left(\tfrac{k+\alpha}{1-\alpha}\right)(n-\lambda-1) \end{aligned}$$

Hence the expected number of destination IP's a scanner access before it can be detected is:

$$E[IP|H_1] = E[Q_{\lambda,n}] = \left(\frac{k+\alpha}{1-\alpha}\right)(n-\lambda-1) \tag{10}$$

Thus from (6) and (10) the min scanning rate is:

$$scan\ rate = \frac{1}{E[N|H_1]} \times \left(\frac{k+\alpha}{1-\alpha}\right)(n-\lambda-1) \tag{11}$$

Closed form expressions for $E[N|H_1]$ is presented in [6]. Substituting for $n-\lambda$ and α we get:

$$scan\ rate = \frac{1}{E[N|H_1]} \times \left(\frac{k+\frac{\theta_1}{\theta_0}}{1-\frac{\theta_0}{\theta_1}}\right)\left(\frac{\ln(\eta_1)}{\ln(\frac{1-\theta_1}{1-\theta_0})} - 1\right) \tag{12}$$