Limitations to Threshold Random Walk Scan Detection and Mitigating Enhancements

Peter Mell
National Institute of Standards and Technology
Gaithersburg, Maryland
peter.mell@nist.gov

Richard Harang
U.S. Army Research Laboratory
Adelphi, Maryland
richard.e.harang.ctr@mail.mil

Abstract—**This paper discusses limitations in one of the most widely cited single source scan detection algorithms: threshold random walk (TRW). If an attacker knows that TRW is being employed, these limitations enable full circumvention allowing undetectable high speed full horizontal and vertical scanning of target networks from a single Internet Protocol address. To mitigate the discovered limitations, we provide 3 enhancements to TRW and analyze the increased cost in computational complexity and memory. Even with these mitigations in place, circumvention is still possible but only through collaborative scanning (something TRW was not designed to detect) with a significant increase in the required level of effort and usage of resources.**

Keywords— Intrusion Detection, Scanning

I. Introduction

Network scans often originate from malicious actors to map out a target network and identify candidate points of penetration [1]. Single source scans originate from a single Internet Protocol (IP) address while collaborative scans originate from a reservoir of multiple cooperating IPs [2]. Horizontal scans probe a single port on many IP addresses. Vertical scans probe many ports on a single IP address. In practice, scans may be a combination of the two.

One of the most cited single source scan detection algorithms is Threshold Random Walk (TRW) [3]. This highly effective algorithm can often detect horizontal scanners after only 4 or 5 connection attempts. It takes advantage of the fact that scanners do not know the mapping of active servers and services to IP addresses and are thus more likely to access inactive services than legitimate users. For each source IP, TRW computes a likelihood ratio (a function of the cumulative sum of successful and failed connection attempts). This likelihood ratio is compared against lower and upper thresholds (set according to false negative and positive tolerances) to label source IPs as either benign or malicious.

However, there are 4 limitations that restrict TRW's effectiveness and applicability in certain environments. The first 3 limitations were acknowledged but not addressed in [3]: 1) attackers can bypass TRW by scanning over non-TCP protocols, 2) attackers can initiate a vertical scan on all ports of a target IP with little or no penalty towards being classified as a scanner, 3) attackers can camouflage their activity by mixing in successful connections with scan attempts. A fourth limitation, and the most impactful, has not been mentioned in prior work: attackers can completely bypass TRW by tricking a detector into categorizing their source IP as benign prior to scanning. This enables full circumvention allowing undetectable high speed full horizontal and vertical scanning of target networks from a single IP address.

These limitations in TRW can be mostly mitigated through simple changes to the algorithm without augmentation with alternate detection approaches. Stateless protocol monitoring can be addressed through providing TRW access to network topology and host/service status information. Vertical scan circumvention can be mitigated through penalizing successive unsuccessful port accesses. And horizontal scan circumvention can be addressed through removing the lower threshold bound and perpetually monitoring source IPs for their level of maliciousness. Even with these mitigations, however, circumvention is still possible through collaborative scanning (something TRW was not designed to detect) but with a significant increase in an attacker's required level of effort and usage of resources.

There is also an increased cost to the defender in terms of time complexity and memory usage. The big-O computational complexity is the same as for TRW, but the constants will normally be much larger. The memory usage increases dramatically as most IP sources are monitored in perpetuity. However, strategies exist to manage the memory growth. Also, we discover that in the original TRW attackers can alter TRW memory usage while in our modified TRW memory consumption is independent of attacker activity.

The remainder of this paper is structured as follows. Section II discusses related work. Section III provides a detailed overview of the TRW algorithm and section IV provides example TRW calculations. Section V enumerates TRW limitations, section VI discusses related mitigation approaches, and section VII analyzed remaining limitations. Section VIII provides a theoretical analysis of the computation complexity and memory requirements of TRW vs. our mitigated version. Section IX concludes.

II. Related Work

There are many approaches to detecting both single source and collaborative scans as discussed in [2]. The subject of our paper, TRW, is one of the most influential prior works and has been used as a fundamental building block in other work.

The following work extends TRW without mention of the limitations addressed in this paper. The work of [4] describes the enhancement of TRW with a severity metric to distinguish reconnaissance scanning from peer-to-peer scanning. The work of [5] examines an extension of the TRW method in which target thresholds for the likelihood ratio are sequentially adapted according to user-defined acceptable risk for a bounded sequence, allowing TRW to converge to a decision in finite time. In [6], TRW is combined with honeypots to reduce false negatives by increasing the weight of connections to honeypot hosts.

However, a couple of papers do mention and, in one case, partially address our stated TRW limitations. [7] cites TRW's dependence on stateful traffic analysis and thus TCP, however, their solution is still TCP centric. It allows monitoring of TCP in backbone environments where state is difficult to determine (e.g., with asymmetric routing). [8] presents a variant of TRW that enables detection of vertical scans but doesn't explicitly discuss or analyze this limitation in the original TRW algorithm. Their approach computes a modified TRW likelihood ratio exclusively for vertical scan detection and a separate one for horizontal scan detection.

III. OVERVIEW OF THRESHOLD RANDOM WALK

TRW is based on the mathematical approach Sequential Hypothesis Testing [9], developed during World War II to reduce sample sizes when testing anti-aircraft gunnery [10].

Conceptually, TRW computes a likelihood ratio, L, for each source IP, s, from the set of all source IPs S. L_s is updated whenever an s attempts to connect to a new distinct target IP address, d. The argument to the likelihood calculation, L_s, is the Boolean set, Y, of the success or failure of connections first initiated by s to each of the IPs in the set of distinct target IPs, D_s. If L_s exceeds a certain threshold, η_1, then s is labeled as a scanner. If L_s falls below a different threshold, η_0, then s is labeled as benign. In either case, the algorithm ceases to monitor s once categorized.

More specifically, TRW records the likelihood ratio $\wedge(Y)$ for each s in S where Y is a list of Boolean values (0=success,1=failure) representing the success or failure of connections to distinct target IPs, D_s. Once calculated, $\wedge(Y)$ is referred to as L_s to specify an application of the ratio to a specific s. H_0 represents the hypothesis that a source IP is benign and H_1 represents the hypothesis that a source IP is a scanner. n represents the number of distinct IP addresses to which s has initiated a connection attempt. With this nomenclature, the likelihood ratio is computed as follows:

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

To calculate the conditional probabilities, TRW uses the following equations.

$$\begin{aligned} \Pr[Y_i = 0|H_0] = \theta_0, \Pr[Y_i = 1|H_0] = 1 - \theta_0 \\ \Pr[Y_i = 0|H_1] = \theta_1, \Pr[Y_i = 1|H_1] = 1 - \theta_1 \end{aligned} \tag{2}$$

θ_0 is the probability that a source is benign given a successful connection while θ_1 is the probability that a source is a scanner given a successful connection. θ_0 and θ_1 must be provided by the user with the constraint that $\theta_0 > \theta_1$. [3] uses θ_0 =.8 and θ_1 =.2 for their experiments.

Lastly, the user must provide the maximum false positive probability, α, and the minimum detection probability, β. [3] uses α=0.01 and β=0.99. Two thresholds η_0 and η_1 are then computed based on α and β.

$$\eta_1 \leftarrow \frac{\beta}{\alpha} \qquad \eta_2 \leftarrow \frac{1-\beta}{1-\alpha} \tag{3}$$

While monitoring the connections of a source IP, s, if at any point $\wedge(Y) \leq \eta_0$ then s is labeled benign whereas if $\wedge(Y) \geq \eta_1$ then s is labeled a scanner. This comparison routine is shown in Figure 1 (copied from [3]).

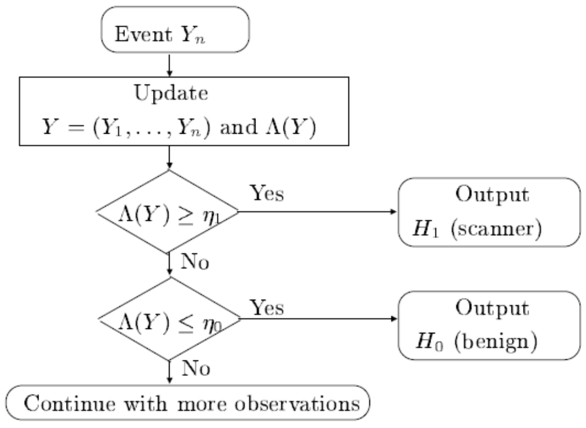

Figure 1. Flow Diagram for Ratio Comparison

The full algorithm that describes how to process lines in a connection log dataset is shown below. We now review the nomenclature: s represents a source IP, d represents a destination IP, D_s represents the distinct target IPs contacted by s, S_s represents the label for s (PENDING, H_0, or H_1), and L_s represents the likelihood ratio for s.

1) Skip the line if S_s is not PENDING (a decision has already made for the remote host s).
2) Determine whether the connection is successful or not. A connection is considered successful if it elicited a SYN ACK.
3) Check whether d already belongs to D_s. If so, skip the next two steps and proceed to the next line.
4) Update D_s with d, and update the likelihood ratio, L_s
5) *If L_s equals* or *exceeds* η_1, set S_s to H_1. If L_s is lower than or equal to η_0, set S_s to H_0.

Algorithm 1. TRW Algorithm from [3]

IV. EXAMPLE TRW CALCULATIONS

To calculate some examples, take the parameters used in [3] where α=0.01, β=0.99, θ_0 =.8, and θ_1 =.2. The lower and upper thresholds are calculated as follows: $\eta_0 = \frac{1-\beta}{1-\alpha} = .0101$ and $\eta_1 = \frac{\beta}{\alpha} = 99.0$. Now consider 3 source IPs. Source 1 initiates a mix of successful and failed connections with a Y for source 1 equal to [1,1,0,1,1,0,1]. Source 2 initiates only failed connections resulting in a Y for source 2 equal to [1,1,1,1]. Source 3 initiates only successful connections resulting in a Y for source 3 equal to [0,0,0,0]. Table 1 provides the likelihood ratios after each connection attempt. Figure 2 displays the results graphically.

Probe	L_1	L_2	L_3
1	4	4	0.2500
2	16	16	0.0625
3	4	64	0.0150
4	16	256*	0.004**
5	64	* permanently labeled scanner; ** permanently labeled benign	
6	16		
7	64		

Table 1. Example Likelihood Ratio Calculations

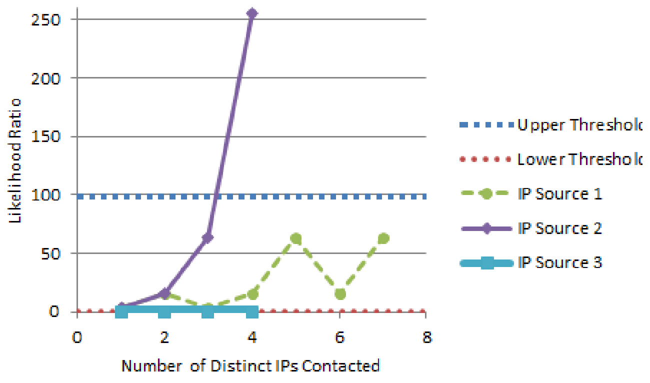

Figure 2. Example Likelihood Ratio Calculations

V. Algorithm Limitations

The TRW algorithm as provided in [3] has several limitations: applicability only to TCP, blindness to vertical scanning, susceptibility to camouflage attacks, and permanent benign classifications.

The first, second, and third limitation were mentioned but not analyzed in [3]. One could argue that the first 2 limitations are merely the expressed scope of [3] and thus it is unnecessary to restate them. However, one point of this paper is that these limitations can be addressed through simple modifications resulting in a greatly enhanced scope for TRW. The fourth limitation, permanent benign classification, enables full circumvention of TRW detection and has not yet been presented in the literature.

A. Applicability only to TCP

The TRW algorithm was design to work exclusively with TCP due to [3] having availability only to TCP logs. From [3], "since the data we have available… consists almost solely of TCP connections, we confine our analysis to detecting TCP scanners." From a technical perspective, TRW is limited to TCP because it monitors for SYN ACK packets, leveraging the stateful nature of TCP to identify established connections.

One obvious way to expand TRW to stateless protocols (e.g., ICMP and UDP) is to monitor for bi-directional communication between IP-port pairs. However, for stateless protocols bidirectional communication is not guaranteed within any particular time frame. Thus, this naïve application of TRW to such protocols may classify valid benign communication as scanning attempts.

B. Blindness to Vertical Scanning

The TRW algorithm was not designed to consider detecting vertical scans (we "do not consider detecting 'vertical' scans of a single host" [3]). The TRW algorithm does not update L_s for each connection attempt from a source IP, s, to a target host, *d*. Instead, it adjusts L_s based only on the outcome of the initial connection to *d*. This means that *s* can scan all ports on *d* and only be penalized for one unsuccessful connection. If *s* is lucky on the first probe and accesses an active service, there is no penalty for the vertical scan. Building on this idea, for a destination IP where an active service is known *a priori*, the attacker can first connect to the known service prior to initiating the vertical scan. In this case, *s* will be able to scan all 65,536 ports while, non-intuitively, having its L_s score moved closer to its being classified as benign. This is true even if all of other ports are inactive. Furthermore, this vertical scan can be repeated periodically and indefinitely without any additional penalty or risk of detection.

An additional impact, not mentioned in previous work, is that this blindness to vertical scanning can decrease the effectiveness of TRW's horizontal scan detection. This happens when the blindness to vertical scanning is combined with evasion approaches using permanent benign classification or camouflage attacks (discussed below).

C. Suceptibility to Camouflage Attacks

An attacker can indefinitely maintain a state of 'PENDING' for an *s* by mixing connections to known services with scanning probes while keeping L_s below η_1. This section describes how to apply this concept to a reservoir of cooperating scanning IPs in order to implement an effective TRW circumvention.

We first provide an equation for the maximum number of unsuccessful connections given a number of successful connections. Next, we derive an equation for the expected number of IP addresses identified given a scanning reservoir of a particular size. Lastly, we simulate the camouflage attack using the network information and parameters provided by [3].

1) Equation for the maximum number of unsuccessful connections given a number of successful connections

Every successful connection decreases $\wedge(Y)$ by $\frac{\theta_1}{\theta_0}$ and every failure increases it by $\frac{\theta_1-1}{\theta_0-1}$ as shown in Equation 1 and Equation 2. If x represents the number of unsuccessful connections and y represents the number of successful connections, then $\wedge(\mathrm{Y}) = \left(\frac{\theta_1}{\theta_0}\right)^y \times \left(\frac{\theta_1-1}{\theta_0-1}\right)^x$. If we set $\wedge(\mathrm{Y}) = \eta_1 < \frac{\beta}{\alpha}$, and substitute in the $\left(\frac{\theta_1}{\theta_0}\right)^y \times \left(\frac{\theta_1-1}{\theta_0-1}\right)^x$ term, then we can use algebraic simplification to isolate x. The equation below then gives the maximum number of unsuccessful connections, x, given a number of successful connections, y.

$$x < \log_{\left(\frac{\theta_1-1}{\theta_0-1}\right)}\left[\frac{\beta}{\alpha} \times \left(\frac{\theta_0}{\theta_1}\right)^y\right] \qquad (4)$$

For the parameters provided in [3], this simplifies to $x < \log_4(99 \times 4^y) = 3.315 + y$ This allows y+3 failed connections prior to *s* being labeled a scanner. In a sense, each successful connection then cancels out each unsuccessful connection for the supplied parameters. However, this is not true in general.

Once an attacker has reached a point at which another failed connection could result in detection, the attacker can migrate to another IP address and continue the scan. To maximize *y*, an attacker can first scan as many known active services as possible. This set can initially be determined beforehand from public information (e.g., web, DNS, and email servers). Once collaborative scanning is underway, any successful probes will reveal active services that can be added to this list to increase the allowable y value of the next source IP to be used. This enables each successive scanning IP to send out a larger number of probes while evading detection.

This idea can be furthered through leveraging TRW's blindness to vertical scanning. After each new connection to a target IP (whether it succeeds or fails), the attacker can vertically scan the target IP with no penalty. Since most IPs will have some port open, this enables the attacker to rapidly increase the size of y for successive scanning IPs; dramatically reducing the required size of the attacker scanning IP reservoir.

While we defer a complete derivation to Appendix A, a stopping time argument combined with standard random walk results shows that, conditional on n services being discovered already, the number Z_i of new services discovered grows in expectation as approximately

$$E[Z_i] = \frac{1}{\gamma-1}\sum_{\tau=1}^{\infty}\Phi\left(\frac{\log\eta_1 - \tau E[X_i]-n}{\sqrt{\tau Var(X_i)}}\right) - \frac{(\log\eta_1 - \log\eta_0)}{\gamma-1},$$

where $\Phi(x)$ is the CDF of the standard Gaussian distribution, growing exponentially for small values of x, X_i is the step increment for each stage of the random walk, and γ is a constant related to the relative weighting of successful and failed connections in the random walk that will be fixed for any given instantiation of the algorithm.

2) Simulation of the camouflage attack

We next employ simulations to examine the impact of this custom scan approach, designed to circumvent TRW, in more realistic (i.e., non-asymptotic) settings. We used Python 2.7.3 with the data from [3] for the LBL and ICSI networks using the supplied TRW parameters. The LBL network had 131836 IP addresses with 5906 active hosts. The ICSI network had 512 IP addresses with 217 active hosts. We make the assumption that each active host has 1 active port from a set of 17 common ports[1] and that hosts are randomly distributed throughout the IP address ranges.

We varied the number of initial services known by the attackers to be active in the networks and determined the necessary size of the attackers' reservoirs in order to scan all IP addresses with the intent of identifying all active hosts. We ran 1000 trials at each data point and provide the results in Figure 3 and Figure 4.

With no prior knowledge of the target network, LBL requires on average 162.3 IPs (with a standard deviation of 13.4) to conduct a full scan. ISCI requires on average 4.3 IPs (with a standard deviation of 1.0). Note the diminishing return on knowledge of existing active services for both the LBL and ICSI scans.

[1] MXToolbox [11] default scan ports: ftp 21, ssh 22, telnet 23, smtp 25, dns 53, http 80, pop3 110, netbios 139, imap 143, ldap 389, https 443, msa-outlook 587, notes 1352, sql server 1433, my sql 3306, remote desktop 3389, and webcache 8080. Mention of MXToolbox is not intended to imply recommendation or endorsement, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

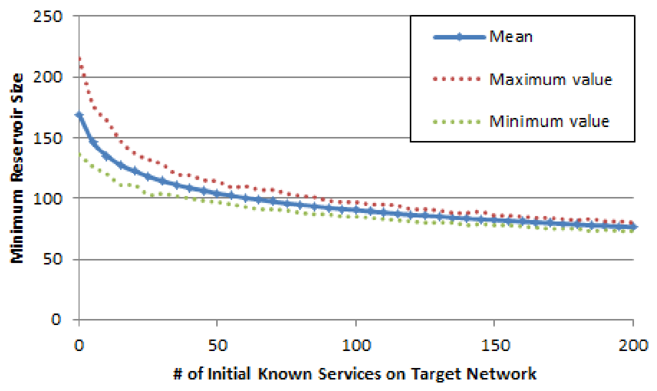

Figure 3. Minimum Reservoir Size for LBL Camouflage Attack

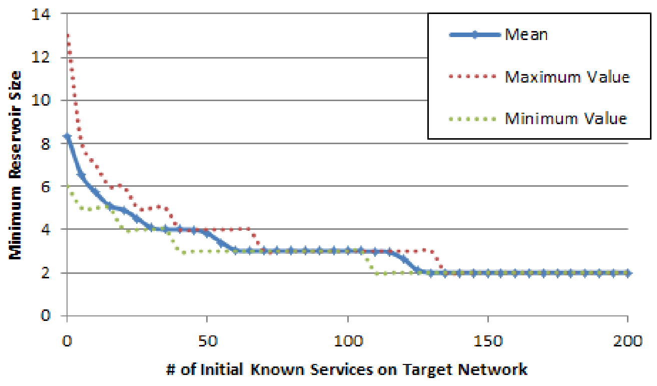

Figure 4. Minimum Reservoir Size for ISCI Camouflage Attack

Figure 5 shows an ensemble of simulated scans on the LBL network, with no prior knowledge, depicting how the scope of the scan increases more than linearly as new scanning IPs are deployed.

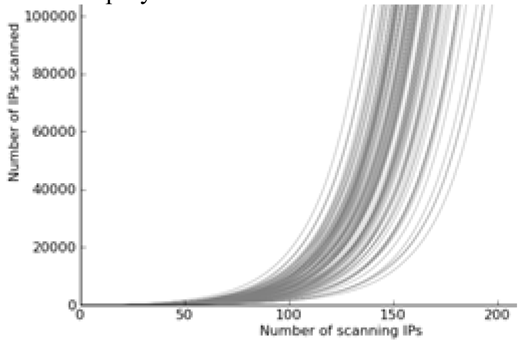

Figure 5. Simulated Camouflage Scans on LBL Network

D. Permanent Benign Classification

TRW permanently classifies source IPs as either scanners or benign. This greatly reduces the time complexity and memory consumption but enables a significant circumvention. Assume that an attacker can *a priori* determine a small number of active and reachable hosts on the target network (e.g., web servers, DNS, and email). In this case the attacker can begin a scan with accesses to these servers. After a small number of accesses, the attacker's source IP will be labeled as 'benign' and ALL further activity will be ignored (see algorithm in Algorithm 1). The attacker may then scan the full target network rapidly and without stealth while evading TRW detection. Reservoirs of cooperating IPs can make this *a priori* knowledge unnecessary as the requisite information can be discovered.

In this section, we first calculate the minimum number of consecutive successful accesses that are required in order for a source IP to be labeled benign. We then derive an estimator for the approximate number of cooperating IP addresses required for an attacker to achieve a "benign" classification. Lastly, we simulate the permanent benign classification attack using the network information and parameters provided by [3].

1) Minimum number of consecutive successful accesses

We now calculate the minimum number of consecutive successful accesses that are required in order for a source IP, *s*, to be labeled benign. s will be labeled benign when $\Lambda(\mathrm{Y}) \leq \eta_0$. Expanding both sides yields $\left(\frac{\theta_1}{\theta_0}\right)^n \leq \frac{1-\beta}{1-\alpha}$ where n is the number of successful connections. Solving for n, $n \geq \log_{\frac{\theta_1}{\theta_0}}\left(\frac{1-\beta}{1-\alpha}\right)$. Thus, once *n* consecutive successful connections have been established, *s* will be labeled benign.

For the parameters used in [3], this simplifies to $n \geq \log_{\frac{.2}{.8}}\left(\frac{1-.99}{1-.01}\right) = 3.315$. Thus, an attacker needs knowledge of 4 legitimate and accessible services to achieve a permanent benign classification.

Now assume that an attacker has no *a priori* knowledge regarding active services on the target network. The attacker can still obtain permanent benign classification through leveraging a reservoir of scanning IPs. The attacker begins with a single IP address and connects to all known services, *K*, (initially this set is empty). The attacker then launches probes and any newly detected services are added to *K*. If a probe is unsuccessful, the attacker vertically scans the target IP on all ports until an active service is found and added to *K*. When the situation arises where the next unsuccessful horizontal scanning probe would trigger detection (as calculated using the formula in the previous section), the attacker moves *K* to a new IP address and commences the same scan procedure. Once *K* is large enough, the next scanning IP used will achieve a permanent benign classification and can scan the target network at will. Note how the blindness to vertical scanning enables us to rapidly build a sufficiently sized *K*.

While closed-form expression for the number of scanning IPs required to achieve this permanent classification are generally intractable, we summarize a general method for estimating this quantity in Appendix B. Using our approach, we can show that on average a pool of 15 addresses will be sufficient for classification as benign in the parameterization of [3].

2) Simulation of permanent benign classification

As above, we next simulated this attack for two realistic non-asymptotic cases using Python 2.7.3 with the data from [3] for the LBL and ICSI networks with the supplied TRW parameters. We use the same assumptions as in the previous simulations except that here the attacker has no *a priori* knowledge. We ran 1000 trials on each network. For the LBL network, we found that the attacker needed a reservoir of on average 22.2 scanning IPs (with a standard deviation of 10.4) to scan the entire network (compared to 15.123 for method-of-moments estimator). This is on average 5939 IPs probed per scanner without being detected. For the ICSI network, the attacker needed on average only 3.2 scanning IPs (with a standard deviation of 0.9) to scan the entire network (versus an estimate of 1.257 from the method of moments estimator, however, note that the independence assumptions underlying the asymptotic analysis are severely violated here). This is on average 160 IPs probed per scanner without being detected.

VI. Mitigations for the Enumerated Limitations

The limitations of applicability only to TCP, blindness to vertical scanning, and permanent benign classifications can be fully addressed through 3 simple modifications to the TRW algorithm. These mitigations are as follows: interfacing with a system state oracle, penalizing successive failed port connections, and removing the lower threshold (η_0).

The limitation of camouflage attacks remains an issue as TRW is fundamentally a single source scan detector and camouflage attacks are in essence collaborative scans. However, implementing these mitigations dramatically increases the resource cost to the attacker when mounting a camouflage attack (analysis in section VII).

A. Interfacing with a System State Oracle

The restriction of TRW to TCP can be overcome through interfacing it with an oracle that provides system topology and state information. In particular, the oracle must provide a list of active IP and port pairs that is updated dynamically as services go up and down. With this information, connections can be graded and used as input to TRW without having to see any return traffic. This enables TRW to monitor non-TCP protocols and eliminates the problems, discussed earlier, with false positives when using stateless protocols.

B. Penalizing Successive Failed Port Connections

The blindness to vertical scanning can be mitigated through adjusting L_s for each connection attempt to each distinct target IP-port pair. This is in contrast to the original TRW algorithm that adjusts L_s only for the initial connection to each distinct target IP. The cost for this modification is increased computation and memory as the list of ports accessed by each source IP must be maintained for every source IP being monitored. Note that we do not perform experimentation demonstrating efficacy of detection since this was done for a related vertical scan solution in [8].

C. Removing the Lower Threshold

The permanent benign classifications limitation can be eliminated through dropping the lower threshold, η_0, and to continuously monitor all source IPs in perpetuity. Source IPs, *s*, are never labeled as benign and thus L_s continuously measures how close *s* is to being labeled a scanner. Using this approach, permanent benign evasion attempts become converted into relatively less effective camouflage attacks (analyzed in section VII).

Removing the lower threshold increases the computational and memory costs as the algorithm must keep track of all source IPs that have ever been seen. To limit this cost, one could time out and delete apparently inactive source IPs. Alternately, one could periodically wipe the algorithm's data store and accept the limitation of only detecting scans performed within the chosen time window.

A lighter weight approach is to restart monitoring a source IP, *s*, immediately after classifying it as benign. This means deleting D_s (the set of all target IPs accessed by *s*) as well as L_s because otherwise the computational penalties of perpetual monitoring would still be experienced. Permanent benign classification is now impossible since IPs labeled benign are immediate reset to a PENDING state. This has a much more modest increase in computation and memory costs. However, this approach opens up another evasion strategy where the attacker alternates launching a small number of probes (few enough for *s* not to be classified as a scanner) followed by connections to enough known services that *s* is labeled benign. The benign classification triggers the deletion of D_s, enabling *s* to launch a few more probes before having to repeat this routine. The attacker can then effectively scan at will from a single IP, as in the permanent benign classification approach, while evading detection at the cost of having to repeatedly issue connections to known active and reachable services.

VII. Remaining Limitations

The proposed mitigations eliminate 3 of the stated TRW limitations: applicability only to TCP, blindness to vertical scanning, and permanent benign classifications. The fourth limitation, susceptibility to camouflage attacks, remains. This is in part due to TRW being in its nature a single source scan detector as opposed to a collaborative scan detector.

However, our mitigations make it much more expensive for an attacker to mount a camouflage attack against TRW to discover all active IPs on a network. To show this, we simulate camouflage scans on the LBL and ICSI networks. We assume that each host has 9 active ports from among the 17 commonly used ports discussed previously which is a worst case assumption for most networks. We chose this high value because additional active ports per target IP will not benefit scans against the original TRW but will benefit scans against the modified TRW. We are trying to show that even with these advantages, scans against the modified TRW require much larger source IP reservoir sizes.

We implement two attacker scenarios, both where the attacker has no prior knowledge of network services: 1) the attacker randomly chooses destination IP/port pairs from hosts not yet fully scanned or identified as active; and 2) the attacker randomly chooses an IP that has not yet been scanned, scans all common ports on this IP until finding an active service, and then repeats the process.

The results of applying these scenarios to both the original TRW and our modified TRW using the mean of 1000 trials are shown in Table 2. Note that we do not simulate the original TRW against attack scenario 1 because it is guaranteed to always do worse than scenario 2 (since TRW allows full vertical scanning without penalty once an IP is probed once). With the original TRW, once a host is probed, the attacker might as well scan all the possible ports and thus no knowledgeable attacker would ever use scenario 1 against the original TRW.

Network	Modified TRW (scenario 1)	Modified TRW (scenario 2)	Original TRW (scenario 2)
LBL	673.86	2810	169.3
ICSI	36.3	105.6	8.4

Table 2. Required Average Reservoir Size

As can be seen, the modified TRW algorithm raises the cost of camouflage attacks and requires the attackers to employ reservoirs of scanning IPs at least 3.98 time larger, regardless of which of the 2 scanning strategies are used.

VIII. Theoretical Complexity Analysis

In this section, we discuss the computational time complexity of TRW and it memory usage. We then compare these results to the modified TRW approach to determine the cost of our proposed mitigations.

A. Computational Complexity of TRW

We provide here a TRW algorithm that has linear time complexity O(*n*) where *n* represents the number of flows being analyzed. We divide our analysis into two parts: the computation time for the oracle and then for the main TRW algorithm.

We design the oracle as a preprocessor to the TRW algorithm and assume that flows arrive to the oracle in chronological order. The oracle reads each flow and appends it to the beginning of a list. It also uses a hash table to map each flow (using the source IP, destination IP, and destination port tuple as the key) to a Boolean value indicating whether or not the flow was part of a successful connection. All flows are initially marked as unsuccessful. When a SYN ACK is received by the oracle's monitor, it looks up the originating flow's tuple in the hash table and marks the flow as successful. The oracle then removes each flow from the end of the list after a certain time threshold. The time threshold is set to the length of time that must elapse in the TRW algorithm for a connection attempt to be marked as unsuccessful when no SYN ACK has yet been received. When a flow is removed from the end of the list, its state is retrieved from the hash table. The flow and state are delivered to the main TRW algorithm and the flow is deleted from both the list and the hash table. This implementation of the oracle works in O(*n*) time where *n* is the number of flows. This is because flow addition and deletion from the list is done in O(1) time and lookup, insertion, and deletion from the hash table is done in O(1) time. Lastly, there will be fewer SYN ACKs monitored than flows (since every SYN ACK must be part of some flow) and thus processing of these packets does not introduce a new term into the computational complexity.

As far as memory, the oracle periodically receives flows and then removes them from memory after a set time threshold. The memory usage for each flow is independent of the other flows being processed and is thus a constant per flow. The oracle thus uses O(*r*) memory where *r* is the current arrival rate of the packets.

We design the main TRW algorithm using the following data structures. We first construct a hash table, H1, that maps each source IP, *s*, to a data structure containing the state of *s* (i.e., pending, scanner, or benign), the likelihood ratio L_s, and a hash table H2. H2 maps each destination IP, *d*, to the value 1. If an entry exists in H2 for *d*, then *s* has communicated with *d*. With these data structures, insertions and deletions of individual source IPs and destination IPs can occur in O(1) time. To process a single flow, the TRW algorithm (shown in Algorithm 1) contains only simple formulas and a constant number of *s* and *d* lookups, insertions, and deletions. Thus, the time complexity of TRW to process all flows is O(n). Note that the actual time complexity is slightly larger as we did not take into account the occasional need to resize the hash tables.

The memory grows relative to the number of distinct IP source/destination pairs where the sources are not marked 'PENDING'. H1 uses O(|S|) memory, not including the H2 sub-tables, since Hash tables can expand linearly to the data being stored and are often resized when reaching 70-80% capacity. If P represents the set of all pairs of communicating IPs, the H2 sub-tables altogether use $O(|(s,d)| \forall (s,d) \in P, s \in S, and\ S_s =' PENDING')$.

B. Computational Complexity of Mitigated TRW

The mitigated oracle receives notices regarding the state (active or inactive) of each of the monitored services represented by tuples of IP address, port number, and protocol name. It can store and update this information in a hash table H3. This takes O(1) time for each notice and consumes O(|D|) memory where D is the set of destination hosts on the monitored network. The ports and protocols are not factored into the time complexity analysis because they are constants (65535 for the number of available ports and 2 for TCP and UDP). Adding new protocols like ICMP simply increases these constants. The oracle can then immediately grade any received flow as either a 'success' or 'failure' relative to the TRW algorithm, without having to wait for any return traffic. This hash lookup from H3 takes O(1) time and thus the oracle takes O(n+|$notices$|) where n is the number of flows. We expect service notifications to be less than n which enables the oracle to operate in O(n) time as with the original TRW oracle. That said, the constant factor could be much greater.

For memory, the O(|D|) usage for our mitigated oracle has to be compared to the O(r) usage for the original oracle. Since |D| can't grow quickly and is under control of the defender while r can be manipulated by the attacker, the mitigated oracle has an advantage in memory management. That said, it must be pointed out that the mitigated TRW oracle may use vastly more memory due to very high constants.

Moving from the oracle to the mitigated TRW algorithm, H2 now must store entries for the tuple destination IP, port number, and protocol name (not just for each source IP). Also, the mitigation of dropping the lower bound n_0 means that many $s \in S$ labeled 'PENDING' would have been labeled benign under the original TRW algorithm. Since there are a constant number of ports and monitored protocols, the time complexity doesn't increase but the constant factors may be much greater.

The memory usage is still $O(|(s,d)| \forall (s,d) \in P, s \in S, and\ S_s =' PENDING')$ but the number of 'PENDING' sources has increased dramatically.

IX. Conclusions

TRW is an effective algorithm for detecting single source scanners. However, it has several limitations that allow for varying degrees of attacker circumvention: applicability only to TCP, blindness to vertical scanning, permanent benign classifications, and susceptibility to camouflage attacks. These limitations allow a single scanning IP to aggressively and repeatedly scan a target network over all protocols and ports without any chance of being detected by TRW. Fortunately, these limitations can be overcome while retaining the fundamental TRW concepts and underlying mathematics through 3 mitigations: interfacing with a system state oracle, penalizing successive failed port connections, and removing the lower threshold. These limitations do not remove the ability to perform camouflage circumvention attacks, but we showed that they significantly increase the resource cost to the attacker. Finally, our mitigated TRW has the same computational complexity as the original TRW but with much higher constants. Memory usage is likewise increased due to the need to keep track of all source IPs. These issues may be addressed through periodically purging old or inactive source IPs from the dataset. We leave it to future work to empirically measure the computational cost and memory of the mitigated algorithm.

Acknowledgment

This research was sponsored by both the U.S. National Institute of Standards and Technology and the Army Research Labs, and was partially accomplished under Army Contract Number W911QX-07-F-0023. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon.

References

[1] S. Panjwani, S. Tan and K. Jarrin, "An experimental evaluation to determine if port scans are precursors to an attack," in *International Conference on Dependable Systems and Networks*, Washington, D.C., 2005.

[2] M. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Surveying Port Scans and Their Detection Methodologies," *The Computer Journal,* vol. 54, no. 10, pp. 1565-1581, 2011.

[3] J. Jung and V. Paxson et al., "Fast portscan detection using sequential hypothesis testing," in *IEEE Symposium on Security and Privacy*, Oakland, CA, 2004.

[4] V. Falletta et al., "Detecting scanners: emperical assessment of a 3G network," *International Journal of Network Security,* vol. 9, no. 2, pp. 143-155, 2009.

[5] X. Chen, "New Sequential Methods for Detecting Portscanners," 2012.

[6] X. Wang et al., "Research for scan detection algorithm of high-speed links based on honeypot," in *Proceedings of IC-NIDC2010*, Beijing, 2010.

[7] A. Sridharan et al., "Connectionless port scan detection on the backbone," in *Performance, Computing, and Communications Conference*, 2006.

[8] L. Aniello et al., "Inter-domain stealthy port scan detection through complex event processing," in *13th European Workshop on Dependable Computing*, 2011.

[9] A. Wald, Sequential Analysis, New York: J. Wiley & Sons, 1947.

[10] T. L. Lai, "Sequential Analysis: Some Classical Problems and New Challenges," *Statistica Sinica,* vol. 11, 2001.

[11] "Supertool," MXToolbox, [Online]. Available: mxtoolbox.com/SuperTool.aspx

Appendix A. Equation for the Expected Number of IP Addresses Identified Given a Scanning Reservoir of a Particular Size

We now build on the equations in [3] to derive an approximate solution for the expected value of the number of active IP addresses identified by the i^{th} scanning IP, showing the exponential growth in the mean value of y with respect to number of scanning IPs.

We consider the asymptotic limit, where the pool of potential IP addresses is effectively infinite and populated at some constant rate such that the probability that any randomly sampled IP address is active is independently and identically p. While this clearly does not hold in practice, it is worth remarking that (as indirectly alluded to in [3]) by considering $Pr[Y_i = 0|H_1] = \theta_1$ to be constant, the TRW method makes an almost identical tacit assumption. We further assume that at least one active service can be obtained from any active IP address. Note that in the following, we concern ourselves with the case H_1; expectations and probabilities are assumed to be taken with respect to that.

As observed in [3], taking the logarithm of the total data likelihood $\Lambda(\bar{Y})$ results in a random walk with absorbing boundaries at $\log \eta_0$ and $\log \eta_1$, and increments

$$X_t = \begin{cases} \log[\theta_0/\theta_1] & \text{with prob } \theta_1 \\ \log[(1-\theta_0)/(1-\theta_1)] & \text{with prob } 1-\theta_1 \end{cases}$$

We place absorbing regions at $S_t \geq \log \eta_1$ and $S_t \leq \log \eta_0$. If N_i services are known in total prior to the beginning of the i^{th} scan, then we set $X_0 = -N_i \log[\theta_0/\theta_1]$; starting the random walk from a lower location as the scanning IP contacts known services prior to beginning its scanning step in order to lower its TRW score.

Conditional on the number of steps that the algorithm required to terminate we can estimate the number of active IP addresses identified on any given scan. We wish to do so conditional on the event that the scan is eventually detected; if it is not detected, and the IP classified as benign, then it will be able to complete the scan of the network without fear of further detection. Retaining the notation in [3], but indexing from 0, we define $S_t = X_0 + \sum_{j=1}^{t} X_j$. If we neglect the lower stopping condition, we then can express the probability that a walk has terminated by the i^{th} step as the probability that the value of that walk exceeds the upper bound. We can then write a minor modification of equation 18 in [3] to include the starting value X_0 as:

$$P\left[\frac{X_0 + \sum_{i=1}^{\tau} X_i - \tau E[X_i]}{\sqrt{\tau Var(X_i)}} \geq \frac{\log \eta_1 - \tau E[X_i]}{\sqrt{\tau Var(X_i)}}\right] \leq P(N_i \leq \tau | X_0),$$

where $Var(X_i) < \infty$ and $0 < E[X_i] < \infty$ are identical to the values given in [3] conditional on H_1 ($\sqrt{\theta_1(1-\theta_1)} \ln\left(\frac{1-\theta_1}{1-\theta_0}\frac{\theta_0}{\theta_1}\right)$ and $(1-\theta_1)\ln\frac{1-\theta_1}{1-\theta_0} + \theta_0 \ln\frac{\theta_1}{\theta_0}$, respectively). If $\theta_1 < 1/2$ then the sequence $\sum_{i=1}^{\tau} X_i$ will form a submartingale: our stopped process is bounded above by $\log \eta_1$ and our increments have finite expectation and variance, and so by the martingale convergence theorem it will converge in the limit to some random variable, even if we neglect the lower absorbing state. We may use this in combination with bounded first and second moments to thus justify a similar appeal to the central limit theorem as in [3], specifically that the normalized sum will ultimately converge in distribution to a standard normal random variable, and write an approximation for $P(N_i \leq \tau)$ as

$$P(N_i \leq \tau | X_0) \approx 1 - \Phi\left(\frac{\log \eta_1 - \tau E[X_i] - X_0^{(i)}}{\sqrt{\tau Var(X_i)}}\right),$$

where, since $N_i \geq 0$, $E[N_i]$ may itself be approximated from the approximate PDF by:

$$E[N_i] = \sum_{\tau=1}^{\infty} \Phi\left(\frac{\log \eta_1 - \tau E[X_i] - X_0^{(i)}}{\sqrt{\tau Var(X_i)}}\right)$$

As in [3] the approximate CDF will generally underestimate the probability of a scan of length τ, and so our expected value will typically be an overestimation of the true value. These results thus represent worst-case scenarios for the defender.

From the work of the previous section, note that in general – defining x as the number of failed probes and y as the number of successful ones – we have $N_i = x_i + y_i$, with a scan terminating when $x - \gamma y \geq \log \eta_1 - \log \eta_0$ for some constant γ. We may substitute and take expectations to find a simple method of moments estimator $\hat{y}_i$ for the number of new services discovered:

$$\hat{y}_i = \frac{E[N_i] - (\log \eta_1 - \log \eta_0)}{\gamma - 1}$$

If we discover $\hat{y}_i$ new services on the i^{th} scan, under the assumption that the scanner is eventually detected, then for iteration $i+1$ we have:

$$X_0^{(i+1)} = \log\left[\frac{(1-\theta_0)}{(1-\theta_1)}\right]\left(\sum_{j=0}^{i} \hat{y}_j\right)$$

A Taylor series expansion for $\Phi(x)$ shows that Φ has an exponential dependence on x for small values of x, and so under the repeated substitutions indicated by the previous two equations, we find that the number of scanned IPs and the number of active IP addresses identified by a scanner is exponential in the number of iterated scans performed.

Restricting our attention to probability of detection on a single iteration, exact expressions for the change in this probability of detection given an attacker's *a priori* knowledge of the network and use of a camouflage attack are tractable in the symmetric case, and again may be approximated in the asymmetric case.If we again assume that (as in [3]), $\log[\theta_0/\theta_1] = -\log[(1-\theta_0)/(1-\theta_1)] = \delta$ (since they have set $\theta_0 + \theta_1 = 1$), we may obtain exact results. Our random walk has symmetric step sizes with asymmetric probabilities, and unique absorbing states above and below. Adjust $\log \eta_1$ and $\log \eta_0$ such that they are integer multiples of δ; as this is a random walk with symmetric step sizes, there are unique values of $\log \eta_i$ that are attainable by the walk given the stopping conditions and this adjustment does not change the result of the analysis.

Note that the value of S_t is strictly bounded, and that the increments of our random walk are finite. We thus are in a position to use basic martingale arguments to show that the probability of a classification of "scanning" satisfies:

$$P_{X_0} = \frac{\rho^{X_0/\delta} - \rho^{(\log\eta_0)/\delta}}{\rho^{(\log\eta_1)/\delta} - \rho^{(\log\eta_0)/\delta}} \qquad (*)$$

where $\rho = \frac{1-\theta}{\theta}$.

To show this, we first observe that the process $Y_t = \rho^{S_t/\delta}$is a martingale, since:

$$E\left[\rho^{S_{t+1}/\delta} | \rho^{S_t/\delta}\right] = \theta\rho^{(S_t+\delta)/\delta} + (1-\theta)\rho^{(S_t-\delta)/\delta} = \rho^{S_t/\delta}$$

Since the time $\tau = \min\{t: S_t = \log\eta_1 \; or \; S_t = \log\eta_0\}$ is trivially a valid stopping time, we obtain by optional stopping that:

$$E[Y_\tau] = E[Y_0] = \rho^{X_0/\delta}$$

And so, denoting $P_{X_0} = P(S_\tau = \log\eta_1 | X_0)$, we can find that for the stopped process:

$$\rho^{X_0/\delta} = P_{X_0}\rho^{(\log\eta_1)/\delta} + \left(1 - P_{X_0}\right)\rho^{(\log\eta_0)/\delta}$$

We solve for $P_{X_0}$to obtain $(*)$. This method allows exact calculation of the approximate values given in [3]. Applying this result to the parameterization of [3], taking $X_0 = 0$, we can then show that $P_{X_0} = \frac{0.25^0 - 0.25^{-4}}{0.25^4 - 0.25^{-4}} \approx 0.9961$, the true probability of detection under H_1, is indeed slightly in excess of β (the minimum chance of detection). However, if the attacker is somehow aware of 2 active hosts on the internal network, we may find $P_{X_0} = \frac{0.25^{-2} - 0.25^{-4}}{0.25^4 - 0.25^{-4}} \approx 0.9376$: lower than the desired β.

If we relax the assumption that our step size is uniform, the problem becomes significantly more complex, and typically does not have closed-form solution. We may find approximate results by application of the generalized Wald identity. For a given parameterization of the scanning problem, find q to satisfy

$$E[e^{qX_i}] = 1$$

And note that given such a value for q, the sequence below forms a martingale with constant expectation e^{X_0}.

$$E[e^{qX_i}]^{-t}e^{qS_t}$$

If as in [3] we assume that the differences $|S_\tau - \log\eta_1|$ and $|S_\tau - \log\eta_0|$ are negligible across all trajectories, we obtain by optional stopping that:

$$e^{X_0} \approx P_{X_0}e^{q\log\eta_1} + \left(1 - P_{X_0}\right)e^{q\log\eta_0}$$

Yielding a modification of the standard random walk result:

$$P_{X_0} \approx \frac{e^{X_0} - \eta_1^q}{\eta_1^q - \eta_0^q} \qquad (**)$$

Note that in the case with symmetric increments our differences $|S_\tau - \log\eta_1|$ and $|S_\tau - \log\eta_0|$ are exactly 0 for appropriated adapted η_i, and we may find immediately that $q = \log\left(\frac{1-\theta}{\theta}\right)$ thus recovering the result $(*)$.

Appendix B. Estimator for the Number of Cooperating IP Addresses

In Appendix A we obtained expressions approximating the probability of detecting a scanner given that the scanner knows some initial number of active hosts. We may use these expressions to estimate the number of cooperative IP addresses required to identify sufficient active hosts that any further scanning IP may obtain a permanent "benign" classification.

First, note that we may find the probability that at least 1 active service is discovered on a given scan simply by exclusion. Recall that if we begin with initial value X_0, then we require $(\log\eta_1 - X_0)/\log[\theta_0/\theta_1]$ failed connections in order to obtain a classification of "scanner". Conditional on the assumption that the scanning IP is ultimately detected, if we have independent probabilities of $1-\theta_1$ on each probe to generate a failed connection, then we can easily find that the probability of generating 0 successful connections is $(1-\theta_1)^{\lceil(\log\eta_1 - X_0)/\log[\theta_0/\theta_1]\rceil}$. The complement thus provides the probability of adding at least 1 IP to the pool of known active internal IP addresses. If we make the assumption that θ_1 is very small, then we may make the simplifying approximation that at most 1 new IP address will be added per scan. Let us denote

$$p_x = (1-\theta_1)^{\lceil(\log\eta_1 - x(\log[(1-\theta_0)/(1-\theta_1)]))/\log[\theta_0/\theta_1]\rceil}.$$

Then note that we may construct a Markov chain $Y_i$denoting the number of discovered active internal IP addresses after i scanning IPs with the following transition rule:

$$P(Y_{i+1} = n | Y_i = n) = p_{Y_i}$$
$$P(Y_{i+1} = n+1 | Y_i = n) = 1 - p_{Y_i}$$

By placing an absorbing state at $Y_i = \lceil -(\log\eta_0)/\log[(1-\theta_0)/(1-\theta_1)]\rceil$ we may numerically evaluate an upper bound on the CDF for the number of required scanning IP addresses, from which an expected value may readily be obtained. Using the parameterization of [3], we may find that $p_x = (0.8)^{4+x}$, and our absorbing state is at $Y_0 = 4$. Our Markov chain transition matrix is thus

$$M = \begin{bmatrix} 0.4096 & 0.5904 & 0 & 0 & 0 \\ 0 & 0.3277 & 0.6723 & 0 & 0 \\ 0 & 0 & 0.2621 & 0.7379 & 0 \\ 0 & 0 & 0 & 0.2097 & 0.7903 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We may find $P(\tau \le i)$ by examining $[M]^i_{[0,4]}$. An expected value for this quantity may be calculated via the usual identity $E[\tau] = \sum_i \left(1 - P(\tau \le i)\right)$. Applying this method to the parameterization in [3], we obtain that $E[\tau] \le 15.288$. Assuming $\theta_1 = 0.9552$ as in the independent approximation for the LNL data, we obtain $E[\tau] \le 200.38$. For $\theta_1 = 0.5762$ as in the ICI data, we obtain $E[\tau] \le 7.047$. It is instructive to compare these results to those of the simulation, highlighting the impact of the dependence between successive scans.

Note that this result represents a weak upper bound to an attacker, since the simplifying assumption is that at most 1 active IP address is identified per scan. For very sparse networks where θ_1 is large and thus $(1-\theta_1)^x$ decreases very rapidly in x, this assumption is a reasonable one and the approximation is useful. For more densely populated networks, we obtain poorer approximations and we may severely overestimate the number of IP addresses an attacker requires as they recover multiple active hosts with each scan. In such cases, the simple geometric estimator $1/(1-P_{X_0})$ may prove more accurate.