Revisiting Network Scanning Detection Using Sequential Hypothesis Testing*

Mansour Alsaleh, P.C. van Oorschot
School of Computer Science
Carleton University, Canada

Abstract

Network scanning is a common, effective technique to search for vulnerable Internet hosts and to explore the topology and trust relationships between hosts in a target network. Given that the purpose of scanning is searching for responsive hosts and network services, behaviour-based scanning detection techniques based on the state of inbound connection attempts remain effective against evasion. Many of today's network environments, however, feature a dynamic and transient nature with several network hosts and services added or stopped (either permanently or temporarily) over time. In this paper, working with recent network traces from two different environments, we re-examine the TRW (Threshold Random Walk) scan detection algorithm and we show that the number of false positives is proportional to the transiency of the offered services. To address the limitations found, we present a modified algorithm (STRW) that utilizes active mapping of network services to take into account benign causes of failed connection attempts. STRW eliminates a significant portion of TRW false positives (e.g., 29% and 77% in two datasets studied).

Keywords: Network scanning detection, port scan, scanning worms, TRW

1 Introduction and Motivation

Network scanning precedes many of today's Internet attacks as it is an initial reconnaissance step for adversaries to locate hosts running vulnerable network services. Each scan event is an attempt to connect to a specific port in a host, either to find out if the host is active or if the port is open and what service it offers. In targeted attacks, adversaries commonly use automated scanning tools (e.g., NMAP [20]) to gather information about the targeted network (e.g., offered services and associated software versions) [21]. Also, many network worms scan various Internet subnets to locate further vulnerable machines to attack [34]. Likewise, scanning is an effective way for a botnet to recruit new bots [16, 17]. Scanning also remains a common technique used to find out if a remote site has a particular security vulnerability in order to use it to host phishing websites [18]. Although numerous network scanning detection approaches have been proposed in literature, very few proposals offer both reasonably accurate and efficient detection. The high false positive rate inherent with these techniques have contributed to few being adopted by IDSs and they are rarely used to automatically block identified scanners.

The majority of proposed scanning detection techniques depends on detecting abnormal network traffic in remote host traffic directed to the local network. Unfortunately, most detection approaches can be evaded easily by adversaries. The only exception appears to be features based on a remote host's successful or failed connection attempts (e.g., [11, 36]) since the objective of scanning is to find open ports and thus it is assumed that the adversary does not know the open ports in the monitored network.

Most scan detection techniques are designed for deployment in an enterprise environment and a controlled environment is often assumed where only limited network services are allowed to operate usually through continuously running servers (e.g., web, mail, and DNS servers). Scan detection schemes based on a remote host's successful or failed connection attempts are ideal for such environments because a failed connection is a good indicator that the remote host lacks knowledge of the available services and thus might be a scanner. That is, assuming a stable network where services and server IP addresses are rarely changed, the probability that benign remotes make failed con-

*June 30, 2011. Contact author: malsaleh@scs.carleton.ca

nections is low. In this case, causes of failed connection attempts from benign sources are usually either misconfigured remote hosts (e.g., setting a wrong destination IP address or port for a service) or some special cases, e.g., web crawlers and proxies (many users could be represented by a single proxy IP address and their traffic through the proxy may look like scanning activity).

On the other hand, in an enterprise with a more relaxed security policy and fewer restrictions on what applications or network services can be used by enterprise workstations (or sometimes with no explicit security policy; e.g., some university networks), new hosts may be added either for the short or long term, host mobility is possible, and there are less restrictions on the client applications or services run by hosts. In enterprise networks which now commonly enable wireless access, the availability of hosts offering services may change rapidly with the device states of the host machines (on/off, hibernated, or disconnected). Furthermore, with dynamic IP configuration, different IP addresses can be assigned to the same physical network port over time. Therefore, transient network services may appear and disappear as devices are powered on/off or physically removed (e.g., in the case of services hosted on mobile devices like laptops). Even in a controlled enterprise environment, some network services might be temporarily unavailable due to maintenance in the hosting servers or network failures.

In many of today's network environments such as these, many benign remote hosts may make failed connection attempts because the services they are trying to connect to, while active in the past, are temporarily unavailable or disabled (because the local host running the service is turned off/sleeping/hibernating, disconnected from the network, or the application running the network service is uninstalled or closed). Thus, if relying on failed connections as evidence of malicious intent in such environments, these issues need to be taken into account for designing a practical network scan detection algorithm, to reduce significantly false positives.

In this paper, we study scanning detection techniques based on a remote host's successful or failed connection attempts in such network environments. In particular, we revisit TRW, a fast online scan detection algorithm [11]. TRW classifies remote hosts as either benign, scanner, or pending according to the ratio of remote host's successful or failed connection attempts in the inbound network traffic within a time window. Based on experimental results from a set of tests on traces from two network environments, we analyse the traffic of remote sources flagged as scanners by TRW showing that a significant number of these sources are false positives (19% - 50%) exhibiting benign behaviour rather than scanning. To overcome this drawback, a new algorithm (STRW) introducing modifications to TRW is proposed which decreases significantly the identified false positives (on average 77% and 29% of TRW false positives are avoided in two datasets collected at different sites). STRW uses network services active mapping method that is used in the LQS scan detection algorithm [2] as a decision oracle while maintaining the method of sequential hypothesis testing used in TRW.

Contributions.

1. STRW: We confirm that TRW was designed for scan detection in a controlled enterprise network environment, identifying several causes of false positives in now common environments of transient nature. Accordingly, we propose a modified algorithm (STRW) for scan detection which takes into account the identified causes of TRW false positives in such environments. STRW shows that the believed hypothesis that behaviour-based network scanning detectors (e.g., TRW) exhibit unsatisfactory performance in residential style network traffic [29] is due to the lack of utilizing information of the characteristics of the monitored environment. In particular, utilizing the monitored network profile to identify benign causes of unsuccessful connection attempts improves significantly the performance of TRW in such environments. We believe that the contribution should be weighed not only by the magnitude of the changes to the original TRW algorithm, but their effect on the algorithm performance in terms of the number of false positives.

2. Empirical Evaluation: We implement STRW as a policy in the Bro IDS [1] (the implementation is available at `http://strw.sourceforge.net/`). We evaluate the detection accuracy of both TRW and STRW on class C and class B networks.

3. Ground Truth Reference for False Positives: We propose using the exposure maps technique [36] as a proxy for ground truth for identifying a lower bound on scan detection false positive results.

Organization. Background on scanning detection techniques is provided in Section 2. We propose and evaluate STRW, a modified TRW algorithm, in Section 3. STRW advantages and limitations relative to other scan detection algorithms are given in Section 3.2. The used datasets and their network environments are given in Section 4.1. We present our evaluation methodology in Section 4. Results and analysis of testing TRW and STRW on each dataset are given in Section 5. Related work is discussed in Section 6. Section 7 concludes the paper.

2 Background

This section gives an overview of the TRW scanning detection algorithm [11], which we subsequently study and test in several experiments. It also briefly describes the exposure maps technique [36] that will be used to build an approximated ground truth for scanning detection false positives. The section also gives a brief review of the LQS [2] scan detection algorithm that STRW utilizes its decision oracle and that we compare STRW.

2.1 Threshold Random Walk (TRW)

The TRW algorithm ([9]; see also [11]) classifies remote hosts contacting a monitored network as either benign hosts or scanners. The distinction criterion is based on the ratio between successful and failed connections attempts initiated by a remote host towards newly visited local addresses in the monitored network. That is, the connection likelihood ratio Λ of each remote host will be updated only when the remote host attempts a connection (whether successful or unsuccessful) with a local host for the first time. Subsequent connection attempts from the remote host to the same local host will not be considered when updating the ratio (whether to the same destination port or to different ones). Therefore, the remote host's Λ will be updated whenever it attempts to connect to a new local address and as long as this remote host has not been classified yet as either benign or scanner (i.e., in the "pending state"). Two parameters are used to update Λ: the estimated probability that a benign remote host initiates a successful connection (θ_0) and the estimated probability that a scanner remote host initiates a successful connection with no prior knowledge of the targeted network (θ_1). According to TRW, If a new connection is successful: $\Lambda = \Lambda \cdot \frac{\theta_1}{\theta_0}$; otherwise, if the new connection fails (i.e., timeout or rejected): $\Lambda = \Lambda \cdot \frac{1-\theta_1}{1-\theta_0}$, where the initial value of Λ is 1. TRW assumes that scanners are more likely than benign hosts to attempt connections to hosts which either do not exist or do not offer the requested service. The updated Λ is compared to an upper threshold and a lower threshold, each of which involves two parameters: the desired detection probability β and the acceptable false positive probability α. If Λ_h of a remote host h becomes less than or equal to the lower threshold $\frac{1-\beta}{1-\alpha}$, the hypothesis that h is benign is accepted. Likewise, if Λ_h becomes greater than or equal to the upper threshold $\frac{\beta}{\alpha}$, the hypothesis that h is a scanner is accepted. However, if h is not flagged as a scanner and a local host initiates a connection to h, h is marked as friendly remote for a period (I_5) during which failed connection attempts from h are not considered. The network security administrator is expected to set these four parameters (θ_0, θ_1, α, β), otherwise the default values are used.

Algorithm 1 gives pseudo-code of the Bro 1.4 implementation of TRW core algorithm [1, 22]. The keyword "def" denotes the default parameter value. Each element in the sets S, B, FC, SC, R, and the table L has a "write-expiry" interval such that each element is removed when the given period of time (I_1, I_2, I_3, I_4, I_5, or I_6) has lapsed since the last time the element was inserted in the set/table. The function `FailedConn(conn)` returns true in the following cases: (a) a RST packet is sent by destination after receiving a SYN packet from source; or (b) a SYN packet or midstream traffic is sent by the source but no SYN-ACK packet is sent by the destination for at least 5 minutes. The function `SuccessfulConn(conn)` returns true when a SYN-ACK packet is sent by the destination.

2.2 Exposure Maps (EM)

In the EM [36, 35] technique, a network exposure map (NEM) is a table of the services offered by a particular network during a training period, built by an automated tool based on monitoring how internal hosts respond to incoming connection attempts. A NEM entry contains three fields {IP address, port, protocol}: internal host IP address, open port number, and the corresponding protocol. To construct the NEM table, all outgoing TCP flows containing SYN-ACK flags are observed and recorded during a training period, with every local host that is seen responding with SYN-ACK flags added to the NEM. For UDP traffic, UDP entries are added when two hosts communicating with the same source and des-

INPUT:
β(def=0.99), α(def=0.01), θ_0(def=0.8), θ_1(def=0.2)
I_1(def=1hr), I_2(def=1hr), I_3(def=0.5hr), I_4(def=0.5hr), I_5(def=0.5hr), I_6(def=0.5hr)
C `//data structure holding current connection information`

OUTPUT:
S (def=$\emptyset$, expires after I_1) `//set of scanners' IP addresses`
B (def=$\emptyset$, expires after I_2) `//set of benign IP addresses`
FC (def=$\emptyset$, expires after I_3) `//set of IP address pairs having failed connection`
SC (def=$\emptyset$, expires after I_4) `//set of IP address pairs having successful connection`
R (def=$\emptyset$, expires after I_5) `//set of friendly remote IP addresses`
L (expires after I_6) `//table of likelihood ratios of remote hosts (i.e., Λ)`

```
1  begin
2      if IsLocalAddress(C.srcIP) then          //a local host, C.srcIP, initiated a connection first to dstIP
3          if C.dstIP ∉ S then
4              add C.dstIP to R
5          end
6          return (False)                                          //since it is outbound connection
7      end
8      if C.srcIP ∈ (S ∪ B ∪ R) then
9          return (False)                      //the remote is already flagged as scanner, benign, or friendly
10     end
11     if FailedConn(C) ∧ ([C.srcIP, C.dstIP] ∉ FC) then
12         add [C.srcIP, C.dstIP] to FC
13         ratio ⇐ (1 − θ1)/(1 − θ0)
14     else if SuccessfulConn(C) ∧ ([C.srcIP, C.dstIP] ∉ SC) then
15         add [C.srcIP, C.dstIP] to SC
16         ratio ⇐ θ1/θ0
17     else return (False)
18     if (an entry in L already exists for C.srcIP) then
19         L[C.srcIP] ⇐ L[C.srcIP] ∗ ratio
20     else
21         add new entry for index C.srcIP into L
22         L[C.srcIP] ⇐ ratio
23     end
24     if L[C.srcIP] > (β/α) then
25         add C.srcIP to S
26         return (True)
27     else if L[C.srcIP] < ((1 − β)/(1 − α)) then
28         add C.srcIP to B
29     end
30 end
```

Algorithm 1: TRW (returns true when a new IP is a scanner). Adapted from [1]

tination port pairs (the first host using a as a source port and b as a destination port, and the second host using b as source and a as destination) are tracked within a short time period. EM is different than Active Mapping technique [26] in that it only collects information about open ports and that it does not require sending any packet.

2.3 LQS

LQS [2] is a lightweight network scan detection algorithm (LQS) that detects scanners from their second connection attempt by default. LQS keeps the state of offered network services over time to evaluate inbound connection attempts. LQS keeps two tables in memory, a table of currently offered services and a table of services that were previously offered and that are not available any more. A threshold-based function of the number of connection attempts to network services that have never been online is computed to determine if a remote host is a scanner.

Relative to TRW, LQS is capable to detect vertical scanning and it is shown to have a better detection accuracy [2]. While TRW requires four consecutive failed connection attempts (with the default parameters) to classify a remote as a scanner, LQS detects

scanners after only two failed connection attempts to two distinct local hosts and thus LQS detects scanners faster from fewer connection attempts.

3 Stateful TRW (STRW)

In this section, we propose modifications (see Algorithm 2) to the TRW algorithm in a new algorithm that we call *Stateful TRW* (STRW). STRW represents a hybrid approach that uses the LQS decision oracle to determine whether a new connection attempt is potentially malicious while maintaining the method of sequential hypothesis testing used in TRW to classify remotes as either scanner or benign. This section also shows the STRW advantages and limitations relative to TRW and LQS algorithms.

3.1 Algorithm

Unlike the two tables in LQS, STRW has a table of network services, NS, for keeping the state of open ports in the monitored network as in lines 8 to 10. The IF statement in lines 11-17 in TRW (Algorithm 1) is replaced by lines 14-20 in the new algorithm (Algorithm 2). First, if the connection is successful and the remote host did not make a successful connection with the local host before (i.e., source/destination IP addresses pair is not in the SC set), the remote host's likelihood ratio is lowered towards being classified as benign. Otherwise, if the targeted network service is not in NS (which indicates implicitly that the current connection attempt is unsuccessful) and the remote host did not make a failed connection attempt with the local host before (i.e., source/destination IP addresses pair is not in the FC set), then the remote host's likelihood ratio will be raised towards being classified as a scanner. Therefore, in the new algorithm, failed connection attempts to previously offered services are now ignored. Consequently, a benign remote host that happens to attempt connecting to any of the network services that are temporary unavailable will not get penalized by TRW for making such failed connection attempts since its likelihood ratio will not be updated.

Entries in NS expire after interval I_7 so that if there is no traffic from the port offering the service indicating the port is still open (e.g., SYN-ACK packet in case of TCP protocol) for a period of I_7, the NS entry will be removed. Choosing an appropriate value for I_7 depends on the properties of the monitored network and the type of the offered network services. I_7 should reflect the approximate duration of inactivity after which a network service is most likely being removed permanently from the monitored network. Ideally, each offered service might have a different expiration time. However, one common parameter for all services is easier to set by administrators given that network services might run on non-standard ports. Using a long expiration time increases false negatives since if a network service is not offered anymore and I_7 is not yet expired, all failed connection attempts to this service will be ignored by STRW. On the other hand, using a short expiration time might increase the number of false positives since failed connection attempts to the service after I_7 expires while the service is temporarily unavailable, will trigger STRW to adjust the corresponding Λ_r towards a scanner.

3.2 Advantages and Limitations Relative to TRW and LQS

Relative to TRW, STRW addresses many of the causes of failed connection attempts from benign sources and therefore it has better detection accuracy in terms of false positives. Unlike TRW, STRW is designed for network environments of transient nature (e.g., wireless and residential usage patterns). While TRW has to wait for the connection state to be known (i.e., whether it is successful or unsuccessful), which gives adversaries a time window to perform further scanning before being detected [2], STRW uses the NS table to immediately decide if the connection attempt is unsuccessful. Thus, STRW is more appropriate for scanning worm detection and it has a better resistance to evasion. Failed connection attempts determined by STRW indicate malicious intent more than those in TRW and thus STRW default parameters can be tightened towards faster detection (see Section 3.3).

Unlike LQS, the sequential hypothesis testing model in STRW provides a theoretical basis for classifying remote hosts. Also, TRW credits remotes that make successful connections by reducing their likelihood ratio towards being classified as benign. While the objective is to avoid penalizing benign hosts that make some failed connection attempts, it enables scanners with a priori knowledge of some available services of the target network to delay detection. On the other hand, LQS can detect vertical scans and it requires a smaller memory footprint than STRW [2].

STRW is similar to TRW and LQS in terms of be-

INPUT: //for an explanation of the use of expiry intervals, see Section 2.1
β(def=0.99), α(def=0.01), θ_0(def=0.8), θ_1(def=0.2)
I_1(def=1hr), I_2(def=1hr), I_3(def=0.5hr), I_4(def=0.5hr),
I_5(def=0.5hr), I_6(def=0.5hr), I_7(def=72hr)
C //data structure holding current connection information

OUTPUT:
S (def=$\emptyset$, expires after I_1) //set of scanners' IP addresses
B (def=$\emptyset$, expires after I_2) //set of benign IP addresses
FC (def=$\emptyset$, expires after I_3) //set of IP addresses pairs having failed connection
SC (def=$\emptyset$, expires after I_4) //set of IP addresses pairs having successful connection
R (def=$\emptyset$, expires after I_5) //set of friendly remote IP addresses
L (expires after I_6) //table of likelihood ratios of remotes (i.e., Λ)
NS (def=$\emptyset$, expires after I_7) //set of offered network services (IP, port)

```
1  begin
2    if IsLocalAddress(C.srcIP) then
3      if C.dstIP ∉ S then
4        add C.dstIP to R                          //a local host initiated a connection first to dstIP
5      end
6      return (False)                                          //since it is outbound connection
7    end
8    if SuccessfulConn(C) then                          //SYN-ACK packet is sent by the destination
9      add [C.dstIP, C.dstPORT] to NS
10   end
11   if C.srcIP ∈ (S ∪ B ∪ R) then
12     return (False)                  //a remote is already flagged as scanner, benign, or friendly
13   end
14   if SuccessfulConn(C) ∧ ([C.srcIP, C.dstIP] ∉ SC) then
15     add [C.srcIP, C.dstIP] to SC
16     ratio ⇐ θ1/θ0
17   else if ([C.dstIP, C.dstPORT] ∉ NS) ∧ ([C.srcIP, C.dstIP] ∉ FC) then
18     add [C.srcIP, C.dstIP] to FC
19     ratio ⇐ (1 − θ1)/(1 − θ0)
20   else return (False)
21   if (an entry in L already exists for C.srcIP) then
22     L[C.srcIP] ⇐ L[C.srcIP] ∗ ratio
23   else
24     add new entry for index C.srcIP into L
25     L[C.srcIP] ⇐ ratio
26   end
27   if L[C.srcIP] > (β/α) then
28     add C.srcIP to S
29     return (True)
30   else if L[C.srcIP] < ((1 − β)/(1 − α)) then
31     add C.srcIP to B
32   end
33 end
```

Algorithm 2: STRW (returns true when a new IP is classified as a scanner)

ing subject to evasion from a scanner with access to a large number of remote hosts (e.g., a botnet) since the scanner can divide the target IP range among these hosts so that each host can scan fewer IP addresses than the STRW threshold to avoid detection (other TRW limitations are discussed by Jung et al. [11]). Relative to TRW and LQS, a training period if needed (see Section 5) represents a limitation in STRW.

STRW should perform better than TRW at the ISP level since externally-accessible network services offered by home users are more likely to be transient (e.g., due to wireless connectivity or home devices being powered on/off). However, bi-directional flows with packet headers must be available for the detector; otherwise, false positives will be generated. In case only flow level traffic collection (e.g., NetFlow) is available, any scan detection mechanism that re-

lies on packet headers will not be applicable (e.g., TRW or STRW). Such limitation could be due to the expected large required storage and computation resources to access packet headers at the ISP level.

Users accessing the Internet wirelessly (e.g., through mobile devices) are more susceptible to lose network connectivity and thus more likely to generate false positives with TRW if any application that requires opening a network port (e.g., some P2P clients and active mode FTP connection) is running in the user device. Therefore, STRW is also suitable for ISPs that provide mobile broadband services (e.g., using HSPA, WCDMA, WiMax and LTE technologies).

3.3 Parameterization

While STRW's detection rate is similar to that obtained by TRW, STRW resulted in a substantially lower FD rate across various parameters values. As discussed earlier, choosing I_7 appropriately depends on the nature of the monitored network. However, given it is usually hard to understand the network behaviour which could also change over time, we recommend setting a large value for I_7 (e.g., $>$ 1 week). The possible increase in the FN number (compared to TRW) is expected to be insignificant because of the very low ratio of open ports to closed ports in all networks. Even with large I_7, the number of entries in NS is limited to the number of offered network services during this period which is expected to be much smaller than the size of the FC or the SC sets for the same period.

We recommend setting $I_3 = I_4 = I_6$ as this helps in decreasing the false positive rate and makes the effect of successful and unsuccessful connection attempts on Λ consistent with θ_0 and θ_1 setting (see Section 5). It is also desirable to set this value higher than the default (e.g., one day vs. 30 min default value) since a higher value detects more stealthy scanners. A possible improvement to STRW is to set I_6 greater than I_1 and I_2 such that, for a remote host r, the corresponding Λ_r continues to be updated even if r is already classified (S and B must be removed from the condition in line 11). If r is already in B and Λ_r exceeds the upper threshold (line 27), r should be first removed from B. Similarly, if r is already in S and Λ_r crosses the lower threshold (line 30), r should be first removed from S.

Since STRW is designed to reduce false positives, we recommend raising the default value for α (e.g., to between 0.05 and 0.10 where the original default was 0.01) to enable detection of more scanners and to flag more benign remote hosts. Setting θ_1 to a lower value than the default, and according to the density of the offered services in the monitored network (i.e., the ratio of open ports to closed ports of all Internet-addressable local hosts) will also improve accuracy. In our conducted experiments, reducing the default θ_1 value reduced slightly the FD rate.

Number of:	Inbound	Outbound
a) Flows (TCP, UDP, and ICMP)	20,238,134	64,956,098
b) TCP connections (flows)	2,285,486	38,619,016
i) Successful TCP connections[1]	61%	9%
ii) Rejected TCP connections[2]	5%	3%
iii) Timed-out TCP connections[3]	34%	88%
c) Remotes initiating TCP connections	478,684	156

Table 1: Dataset statistics of Class C network (dataset of March 15-28, 2009; (i), (ii), and (iii) are percentage of b)

4 Datasets and Evaluation Methodology

This section first describes the datasets and the associated network environments. It then describes our methodology in obtaining approximated ground truth for identifying false positives in TRW and STRW detection results.

4.1 Datasets and Network Environments

Class C dataset. We used a network trace of packet headers collected at two class C subnets of a university with 156 Internet-addressable IP addresses in total (see Table 1). Both subnets are located on the same network switch (primarily client network with no DNS server) where we gathered the trace over the period March 12-28, 2009. We used the dataset for March 15-28 for all the experiments except the second set in Section 3 where we used the full trace. The size of the March 15-28 dataset is 381 gigabytes (on average 27 GB/day); 0.06% of the packets were

[1]SYN-ACK packet is sent by the destination.

[2]RST packet is sent by the destination after receiving SYN packet from the source.

[3]SYN packet or midstream traffic is sent by the source but no SYN-ACK packet is sent by the destination for at least 5 minutes.

dropped by the tracing program. All the IP addresses were populated (originated outbound traffic during the period).

The network has an open security policy in both subnets, with no restrictions on the network services the clients can offer. Although the assigned IP address of each network access point did not change during the capture period, each user can set up a router at their network access point to connect multiple machines (though most of the users did not have routers and thus only few NATs exist). For simplicity, we only study TCP traffic. We conducted the experiments off-line using the Bro 1.4 NIDS [1].

Using a signature-based detection approach (which is not central to the contributions in this paper), Table 2 gives a summary of the observed protocols running in the open TCP ports in the Class C dataset. Since we collected only the maximum TCP/IP header size (the first 96 bytes of the packet), only protocol signatures located in the first bytes of the TCP payload data for packets with shorter than maximum header size are identified. Note that this limits a detailed network behaviour analysis. We do not rely on the destination port number to identify the corresponding protocol. Only the first initiated TCP connection to each of these open ports is analyzed and thus it is possible that other protocols not captured in Table 2 are used later through these open ports. Moreover, various protocols can be tunneled over some of the reported protocols in Table 2 (e.g., over HTTP or SSL). For instance, similar to web traffic, some protocols including P2P use HTTP requests for file transfer [12]. Table 2 indicates that the majority of the open ports in this dataset belong to what appear as network services used by MSN, HTTP, and P2P data transfer protocols.

Class B dataset. The dataset contains packet headers (554 gigabytes) and it is gathered from a class B university network (a university different than the one in Section 4.1) over the period February 1-11, 2010. The network environment is controlled in that the majority of the users can not install any software. No software that offers a network service is installed in any workstation by default; however, some workstations have open ports (45% of TCP outbound connections failed vs. 95% of inbound connections failed). In both datasets, IP addresses are not shuffled between access points (e.g., by a DHCP server) and there is no DNS servers.

For the Class B dataset, the extracted network services table contains 498 open TCP ports. The open ports fall into the following categories: (a) 184 are web servers offering both http and https (most are printers offering web interface for configuration); (b) 68 FTP sessions (c) 46 SSH servers; (d) 4 DNS servers; (e) 2 SMTP servers; and (f) 185 various other services mostly on ephemeral ports.

4.2 Methodology

In this section, we first briefly review related work on the acquisition of ground truth of scanners in a given dataset, and then present our evaluation methodology for the TRW and STRW algorithms, including a new mechanism to establish ground truth of false positives.

4.2.1 Ground Truth of Scanners (Background)

Acquisition of ground truth data is needed to establish confidence in the results of any scan detection algorithm; the ground truth is an authoritative reference for determining TP and FP rates. While labeled datasets are typically used to evaluate the performance of an IDS technique, labeling a dataset with scanning activities is a far less scientific proposition, as is not conventional since intrusion signatures can not reliably assure the scanning intent. Also, generating synthetic scanning activity does not reflect the various possible port scanning mechanisms.

While it is possible to use IDSs to identify some scanners with network traffic containing known signatures for worms or network attacks, it is not always the case that network traffic originated from a scanner will contain malicious payloads. Carrying a malicious payload in the first packet of the connection is more likely when adversaries scan UDP ports while a scanner targeting TCP ports that can not make a successful connection (i.e., could not find an open port) is expected to send only SYN packets (e.g., an HTTP scanning worm). Therefore, IP addresses of sources sending malicious code might represent only an insignificant portion of the number of all scanners contacting the monitored network.

A possible way to get a measure of ground truth is to monitor (in a type of baseline training) remote hosts' network traffic over moderate periods of time (e.g., a few days) and flag those in which the majority of their flows are unsuccessful. For example, as a way to identify scanners Jung et al. [11] used the observation that for a given remote host (in their datasets),

Protocol	Ports count	Description
MSN	2,826	MSN Messenger file transfer, audio, and video protocols
BitTorrent	114	P2P file sharing protocol
HTTP	137	Hypertext Transfer Protocol
NBSS	68	NetBIOS Session Service
SMB	43	Server Message Block (file or print sharing)
DCE/RPC	37	Distributed Computing Environment/Remote Procedure Calls
VoIP	12	Only Skype protocol is detected
SSL	10	Secure Sockets Layer
Others (known)	111	GTP, NDPS, TDS, UCP, OPSI, eDonkey, RMI, SLSK, COPS, DAAP, DIS, DISTCC, ENTTEC, Etheric, giFT, H1, LDAP, MSMMS,PCAP, PPTP, TNS, YMSG, Zebra, RTSP, CAST, Diameter, IDAP, X.25
Others (unknown)	1,603	Most packet payloads in binary format
Total	4,961	

Table 2: Protocols used in the offered services in the class C dataset

the percentage of the local hosts for which the connection attempt with the remote host failed is either very close to 0% or 100%. They then set a threshold of 80% or more failed connection attempts to identify scanners.

While this heuristic might be true for a large portion of scanners, it could also include a significant number of benign sources making repeated unsuccessful connections because of the unavailability of the contacted services as discussed in Section 1. Moreover, stealthy scanners who contacted some legitimately offered services in the monitored network might not exceed this threshold even over a relatively long period of monitoring and hence will not be flagged as possible scanners. On the other hand, it is hard to conclude with certainty that a source with only a single or very few connection attempts, the majority of them unsuccessful, is a scanner. Although there is no 100% error-free scan detection technique in the literature to date, comparing the detection results to other algorithms' results can also help in validating scan detection results.

4.2.2 Evaluation Methodology

The traditional metrics for measuring the performance of a scan detection algorithm are the true positive (TP) rate and false positive (FP) rate. The TP rate is the proportion of the distinct IP addresses of scanners that were correctly reported as scanners (TP rate $= TP/(TP + FN)$, where FN is the number of false negatives; i.e., distinct IP addresses of scanners that were erroneously classified as benign). The FP rate is the proportion of the distinct IP addresses of benign sources that were erroneously classified as being scanners (FP rate $= FP/(FP+TN)$, where TN is the number of true negatives; i.e., distinct benign source IP addresses that were correctly classified as non-scanners). Notice that a remote host is counted only once in computing TP, FN, FP and TN regardless of the number of connection attempts initiated by the remote.

Herein we propose using the EM technique to evaluate scan detection performance only from the perspective of number of discovered false positives. That is, we use the EM technique to identify most of the benign remote hosts and thus if TRW classifies any of them as a scanner it is counted as a false positive. In particular, we give an approximate measure of the false discovery rate (FD rate $= FP/(FP + TP)$) for the TRW algorithm, i.e., the proportion of the reported positive by TRW that are false positives. Given that, regardless of the detector performance, the FP rate (FP rate $= FP/(FP + TN)$) of a scan detector is expected to be small (e.g., less than 1%) since the number of non-scanner remotes (i.e., TN) is significantly larger than the number of scanners (i.e., TP) [2] in a given dataset, we argue that the FP rate is not a significant metric in determining the detection accuracy of scan detectors. Rather, the count of false positives and FD rate are better measures of the scan detector accuracy and the volume of false alarms in a given time window.

As discussed in Section 3, the modifications made in STRW aim to reduce the proportion of the actual negatives that are erroneously reported as positive by TRW and not to increase the number of hits (i.e., true positives). Therefore, ground truth of false negatives

is not necessary. We choose not to use the proposed method to establish an estimate ground truth of scanners in [2] as our proposed method gives a more solid ground truth for identifying false positives, which is what we require for the evaluation.

We have implemented the EM technique in the Bro language for validating TRW results. First, we used EM to enumerate network services offered by the local hosts in the dataset (as described in Section 2.2) and also the services availability based on timestamps. Knowing previously all actively responding ports in the network (i.e., by creating the NEM table as in Section 2.2), we traverse the dataset flagging any remote host initiating a TCP connection to a local IP/port not in the offered service table as a possible scanner; we call this list of source IP addresses A.

While the list might contain benign sources, it will contain all scanners excluding those who probed only open ports. However, given that a network service in the NEM table might not be offered before being added to the table, it is possible that some remote hosts making failed connection attempts to this service before being added to NEM are actually scanners. Nevertheless, the probability $Pr(FN)$ of a specific scanning IP address coincidently attempting connections with only the network services in the NEM table is very low; $Pr(FN) = \prod_{i=1}^{n} \frac{no.\ of\ entries\ in\ NEM}{2^{16} \times\ no.\ local\ hosts}$ where n is the number of connection attempts from the remote host (assuming that all ports are scanned with equal probability). Therefore, remote hosts not in the list A that are flagged as scanners by a scan detection mechanism are false positives with a very high probability.

Using the default values of TRW parameters as in the Bro 1.4 implementation [1], any remote host IP address in the scanner's set will be removed as a member of the set after a one hour time period even if there is scanning activity from the same remote host within this hour, and likewise for the sources in the benign set. From a preliminary experiment using the default parameter values in the Bro implementation (see Algorithm 1), it was not clear how to deal with a continually changing list of scanners. Specifically, after observing all the IP addresses added to the scanner list over time and likewise for the benign list (sets S and B in Algorithm 1), about one third of the IP addresses in the scanner list also existed in the benign sources list which suggests that about one third of the marked remote hosts exhibit both malicious and benign behaviors. While it is possible that some remote hosts might change state from benign to a scanner (e.g., compromised by a worm) or vice versa, many of these remote hosts were marked by TRW as benign and then scanner (or the opposite) several times during the network trace time frame.

To overcome this issue, we made both lists permanent as in the original TRW implementation [9] (i.e., expiration times I_1 and I_2 are removed). We also set expiration time I_6 to one day rather than the default value (30 minutes) to be able to detect more stealthy scanners (i.e., those which do not exceed TRW threshold within the default time window). However, TRW requires much more memory (e.g., more than 1GB of RAM with the particular dataset used) if all the write-expiry intervals in Algorithm 1 (i.e., $I_1 - I_6$ where the element is deleted when the given amount of time has lapsed since the last time the element was inserted in the set or the table) are removed as in the original TRW implementation. Therefore, the Bro implementation [1] with the default write-expiry intervals is more memory-efficient, but decreases detection accuracy.

We then performed several experiments using different values for TRW parameters. In each experiment we compare the list S of scanners flagged by TRW to the possible scanners list A we created earlier. It is expected that TRW will detect a subset of A, however, IP addresses in S and not in A represent remote hosts possibly misclassified as scanners by TRW. These IP addresses made connection attempts to only IP/port pairs that exist in the NEM table, and thus it is unlikely that they were performing network scanning. Therefore, these IP addresses represent the subset of TRW detected scanners that are false positives. We call this set TRW approximated ground truth aGT_{FP}. If at least one of the failed connection attempts that TRW considers in classifying a remote host as a scanner is destined to an IP/port pair in the NEM table, this is also a possible false positive. We call this set $TRW\ aGT^*_{FP}$ ($aGT_{FP} \subseteq aGT^*_{FP}$). For the experiments in Section 4 we use $TRW aGT_{FP}$ set since the probability of having a scanner in this set is lower than for $TRW\ aGT^*_{FP}$.

5 Evaluation

This section reports the results of testing TRW and STRW on both datasets. False discovery rate (FD rate $= FP/(FP+TP)$) is given for each experiment.

5.1 Class C Dataset

TRW. The NEM table of the network trace contains 4961 TCP ports offered by 78 unique local IP addresses. While there are 4250 distinct open TCP ports, there are only 5 non-ephemeral ports (80, 139, 443, 445, 1024) having only 19 distinct entries in the network offered services table. The most frequent open port is port 80 with 13 entries and then port 1412 with 6 entries. Over 60% of the table entries are from the dynamic or private ports (i.e., from 49152 through 65535). While there are 19 hosts with over 100 open ports for each, the maximum number of open ports per host over the network traffic capture period is 627 and the average number is 32 per host. A local IP address with a high number of open ports is either used by one machine running network services that use different ports at different times, or assigned to different machines during the capture period. $Pr(FN) = \prod_{i=1}^{n} \frac{4961}{2^{16} \times 156}$ where n is the number of connection attempts from the remote host (e.g., $Pr = 0.0005$ for $n = 1$).

Table 3 lists the results of ten selected TRW experiments with values for TRW parameters as shown (default input parameter values of Algorithm 1 are used except as noted). The number of false positives represents the size of $TRWaGT_{FP}$ (as defined in Section 4.2). The first observation is the dramatic change in the number of detected scanners and benign sources with only minor change to one of the TRW parameters. For example, the number of detected scanners triple by changing the α from the default value, 0.01, to 0.10. The parameters θ_0 and θ_1 affect how a remote host's likelihood ratio gets updated. α and β affect the upper and lower thresholds for flagging a remote host as either scanner or benign. The value of β (the desired detection probability) is typically set very high since the objective is to detect the scanners [11]. However, while it is always desired to have a lower false positive rate, increasing α increases the number of detected scanners as well.

To the best of our knowledge, there is no publicly available criterion specifying how to choose appropriate values in TRW to get satisfactory detection results. In addition, the TRW implementation (as in the Bro IDS [1]) contains 'write expire' intervals for the sets S, B, FC, SC, R and the table Λ (as in Algorithm 1) so that an entry is deleted from the set/table if not inserted again within a predefined time window. Such intervals, while helpful to avoid running out of memory especially with high network traffic volume, impose a new challenge for network administrators: choosing appropriate values according to the network nature or environment. Another observation is the low number of remote hosts classified by TRW as either scanners or benign. In all the experiments, TRW classified less than 20% of remote hosts initiating connection attempts leaving over 80% unclassified, pending further events.

The first experiment uses the default TRW parameters as in the first row in Table 3 which shows that about half of detected scanners are false positives (relative to $TRWaGT_{FP}$). 2,383 IP addresses are marked by TRW as scanners but they were not in the set A and thus these remote hosts only attempted connections to the ports that were observed at one time to offer a service. The fact that some of their connections failed are due to temporarily unavailable services and not because they are scanning the network. Upon tracking down several local hosts of the network services causing failed connections, we found the typical scenario occurs after the local host gets disconnected from the network for some time and thus its corresponding services become inaccessible.

As discussed in Section 1, several possible factors can cause a local host to be disconnected from the network (e.g., restarted after a system update or turned off). If a remote host is not in the TRW benign list and happens to attempt connecting to any of the services in this temporarily unavailable local host, then it will get penalized by TRW for making failed connection and its likelihood ratio will be raised towards being classified as a scanner. In some other cases, the local host was reachable but some of its network services were not responding. The most likely scenarios are that either the application running the network service was closed or uninstalled. Consequently, when the remote host made unsuccessful connection attempts, such that its likelihood ratio exceeded the upper bound, it was classified as a scanner by TRW. In most cases, however, we found that those network services causing failed connections became active again as a result of either the users turning the corresponding hosts on, reconnecting to the network, or restarting the related applications.

Since θ_1 models the probability a scanner remote host with no prior knowledge of the targeted network initiates a successful connection, we reduced the TRW default value of θ_1 to 0.05 in the second experiment. As expected, this change slightly increased the number of reported scanners. However, the number of reported benign sources increased by about 4 times since now only two consecutive successful connections

Experiment number	Non-default parameters - I_1 and I_2 are not used - $I_6 = 24hr$	no. of benign remotes	no. of remote scanners	no. of FP (distinct scanners)	FD rate (distinct scanners)
1	None	9,734	5,071	2,383	47%
2	$\theta_1 = 0.05$	39,290	7,948	3,417	42%
3	$\theta_1 = 0.02$	39,886	7,881	3,373	43%
4	$\theta_0 = 0.6$	7,146	2,132	1,064	50%
5	$\theta_0 = 0.6$, $\theta_1 = 0.05$	39,396	2,626	1,265	48%
6	$\alpha = 0.10$	9,581	16,525	6,437	39%
7	$\alpha = 0.10$, $\theta_1 = 0.05$	39,142	15,839	6,123	39%
8	$\alpha = 0.10$, $\theta_1 = 0.4$	4,015	8,947	3,922	44%
9	$\alpha = 0.10$, $I_3 = 6hr$, $I_4 = 6hr$	3,126	5,853	1,729	30%
10	$I_3 = 24hr$, $I_4 = 24hr$	1,350	57	11	19%

Table 3: Experimental results using TRW showing FP (false positives) and FD rate (false discovery rate[4]).

are required by TRW to classify a source as benign. 3,417 remotes in the TRW scanner list S are not in set A and thus the false positives represents 42% of the number of detected scanners. Similar results are obtained by changing θ_1 from 0.05 to 0.02 in the third experiment.

Our analysis of the network traffic shows the transient nature of the offered services which suggests that the probability of a benign remote host initiating a failed connection is actually high compared to the intended probability in TRW default settings. Therefore, to decrease the number of false positives, we decreased θ_0, the estimated probability a benign remote host initiates a successful connection, to 0.6 in the fourth experiment. However, although the number of scanners decreased to less than half that in the first experiment, about 50% of the detected scanners are false positives. Adjusting θ_1 and α in experiment 5, 6, 7, and 8 did not help either in decreasing the number of false positives.

In experiment 9 and 10, we increased the time window for keeping the status of successful and unsuccessful connections initiated by remote hosts. The FD rate in experiment 9 decreased to about 30%. Similar configuration in experiment 10 also reduced the FD rate to 19%, however, the number of reported scanners diminished to 57 only. The reduction in the number of false positives is because of the increase in the expiry time (I_3 and I_4) for the entries in the sets FC and SC, increasing the duration used for keeping the state of connections for inbound traffic. Knowing that a remote host r has unsuccessful connection with a local host l in the past and keeping this information long enough prevents TRW from repenalizing r for making another unsuccessful connection with the same host, l. This would have happened if the previous failed connection was not found in the set FC (as in line 11 in Algorithm 1) and Λ_r will be raised towards being classified as a scanner, which explains the slight reduction in the number of false positives in the experiments 9 and 10. On the other hand, knowing that r connected successfully to l in the past and keeping this information in the set SC will not prevent TRW from penalizing r if r happens to contact l while either the network service in l or the host itself is temporarily unavailable (as in line 14). Considering if the current unsuccessful connection is in SC first before increasing r's likelihood ratio (towards being classified as a scanner) will help in reducing the number of false positives.

Choosing an appropriate expiry time for the entries in the sets FC and SC is challenging since it depends on how often client applications connect to the network services. Keeping the entries for too long consumes considerable memory since the status of every inbound connection, whether successful or not, together with the source and destination addresses need to be stored. Also, choosing long values for I_1 and I_2 will not reflect the possible change of state in remotes from benign to scanner and vice versa since the algorithm does not change the likelihood ratio of the classified remote hosts (line 8 and 9).

STRW. The NS table in STRW differs from the one used in obtaining $TRWaGT_{FP}$ as defined in Section 4.2. Notice that while the table of network services used in $TRWaGT_{FP}$ includes all services offered at any time during the dataset capture period,

[4]Note that the reported false discovery rate (FD rate) in Table 3 is different than false positive rate (FP rate) as explained in Section 4.2.

an instance of NS contains only the active services during I_7 time window. For the dataset described in Section 4.1, we first set I_7 to three days (mainly to take into account machines possibly turned off on weekends). STRW1 in Table 4 shows the results of repeating the same previous ten experiments on STRW with I_7 set to three days. Relative to $TRWaGT_{FP}$, the number of false positives decreased in all experiments compared to TRW by 35% on average. While this is a substantial improvement, FD rate is still high (an average of 34% of detected scanners by STRW are false positives). Although the number of scanners is also decreased by 25% on average, the reduction is more than the difference in the number of false positives for all the experiments. That is, K = (no. of scanners in TRW − no. of scanners in STRW) − (no. of FP in TRW − no. of FP in STRW) > 0. The difference, K (an average of 673 scanners per experiment), represents additional false positives in TRW (unconsidered in Tables 3 and 4) that can be added to our original discovered false positives (i.e., $TRW\ aGT^*_{FP} - TRWaGT_{FP}$, as in Section 4.2).

These remote hosts made failed connection attempts with both IP/port pairs both in the NS table and IP/port pairs not in the NS table (i.e., open and closed ports in the monitored network), but unlike TRW, the number of connection attempts to distinct IP/port pairs not in the NS table is not high enough to trigger STRW to report them as scanners. On the other hand, the number of remote hosts declared to be benign is more in STRW than TRW in every experiment by 3% on average. This represents remote hosts that made enough successful connection attempts to be flagged as benign but also made failed connection attempts to NS entries which made TRW report them as scanners or pending.

There are two reasons that STRW1 FD rate is still not low: (i) services temporarily unavailable for more than three days increase the number of false positives; and (ii) time is required for the NS table to get populated with available services. Hence, it is preferable to increase I_7 and to run STRW for a few days (i.e., for a period of I_7) prior to considering its scan detection results. Consequently, in STRW2, I_7 is set to one week and STRW is run three days before considering the scan detection data (March 12-14). After the first three days (by beginning of March 15th), the NS table was populated by 1097 network services and all other variables were set back to their default values. Table 4 shows the results of repeating previous experiments using STRW2. The average number of false positives in STRW2 decreased by 77% compared to TRW in Table 3. The FD rate in STRW2 (an average of 16%) is also substantially less than TRW. In addition, the number of scanners decreased by an average 43% compared to TRW. On average, $K = 700$ remote hosts declared to be scanners per experiment (i.e., additional false positives in TRW to be added to FD rate in Table 3). As discussed in Section 3.3, the required memory for the NS table was insignificant (less than 3000 entries of IP/port pairs at any given time) as it is bounded by the number of offered network services.

5.2 Class B Dataset

With the default TRW settings, a remote host making consecutive failed attempts to 4 or more internal hosts in the target network will be reported as a scanner. Hence, according to $TRWaGT_{FP}$ (see Section 4.2), having less than 4 internal hosts offering network services implies that there will be no false positives using TRW default settings. However, benign remote hosts which make failed connection attempts to temporarily unavailable services in addition to some non-existing services (at least one but less than 4) might be mistakenly flagged as scanners by TRW (i.e., $TRW\ aGT^*_{FP}$; see Section 4.2) even if there are less than 4 offered services. Therefore, although the probability of having false positives in TRW increases with more network services, STRW is expected to perform better than TRW even with few offered services.

Considering $TRWaGT_{FP}$ only, the FD rate for TRW was less than 1% for all the 10 TRW experiments corresponding to those in Section 5.1. In all the experiments, both STRW1 and STRW2 had no false positives. However, the number of scanners in STRW1 and STRW2 was less than TRW due to some failed connection attempts that were made to temporarily unavailable network services and thus STRW did not consider them in updating the likelihood ratio of the corresponding remotes towards being classified as scanners. For $TRW\ aGT^*_{FP}$, the highest FD rate for TRW is in experiment 8 (24%) in which it requires only two consecutive failed connection attempts to classify a remote host as a scanner. Using STRW2 in this second dataset, on average 94% of TRW false positives in all experiments are eliminated. All the scanners detected by STRW are also detected by TRW.

While the detection accuracy of STRW over TRW in this dataset is not as substantial as in the previous

Exp.	Non-default parameters - I_1 and I_2 are not used - $I_6 = 24hr$	no. of benign remotes		no. of remote scanners		no. of FP (distinct scanners)		FD rate (distinct scanners)	
		STRW1	STRW2	STRW1	STRW2	STRW1	STRW2	STRW1	STRW2
1	None	9,993	9,999	4,290	3,143	1,757	633	41%	20%
2	$\theta_1 = 0.05$	39,682	39,708	6,525	4,872	2,496	881	38%	18%
3	$\theta_1 = 0.02$	40,002	40,024	6,513	4,862	2,494	881	38%	18%
4	$\theta_0 = 0.6$	7,203	7,206	1,866	1,333	814	291	44%	22%
5	$\theta_0 = 0.6$, $\theta_1 = 0.05$	39,731	39,751	2,323	1,688	984	360	42%	21%
6	$\alpha = 0.10$	9,938	9,946	12,122	9,592	3,785	1,300	31%	14%
7	$\alpha = 0.10$, $\theta_1 = 0.05$	39,601	39,628	11,977	9,470	3,762	1,299	31%	14%
8	$\alpha = 0.10$, $\theta_1 = 0.4$	4,379	4,383	6,695	5,018	2,519	883	38%	18%
9	$\alpha = 0.10$, $I_3, I_4 = 6hr$	3,301	3,304	3,896	3,186	1,195	501	31%	16%
10	$I_3 = 24hr$, $I_4 = 24hr$	1,405	1,404	22	20	2	0	9%	0%

Table 4: STRW experimental results. FP denotes false positive and FD rate denotes false discovery rate[5].

dataset, STRW remains a better choice in such (more static) environments for several reasons. First, sudden changes in the availability of the network services do not add false positives. Even main servers in enterprise environments may occasionally lose network connectivity due to various causes: software patches, hardware upgrades, power outage and network devices failures. Second, taking into account that network devices (e.g., routers and switches) and IDS detectors might skip packets that cannot be processed in real time in some congested subnets of the network, some connection attempts might erroneously be interpreted as unsuccessful (e.g., uncaptured SYN-ACK packets) which will cause false positives in TRW.

6 Related Work

Various approaches for detecting network scanning have been proposed. The majority require only IP packet headers. Different subsets of header contents are analysed by different mechanisms to infer scanning activity. While the analysis usually depends on statistical models, there are machine learning based methods [3, 33, 27] and visual-based mechanisms [14, 4, 32]. A few proposals correlate remote scanners to detect coordinated scans [37, 6]. In addition, several scanning worm detection techniques involve ways to detect port scanning (e.g., the rate at which hosts initiate connections to newly visited local hosts) focusing on infected machines in the local network [30, 25, 34, 10]. A few approaches focus on backbone traffic such as ISPs where usually only uni-directional flows are available [7, 28].

NSM IDS [8] is one of the early approaches which only looks at the destination IP addresses contacted by a remote host. NSM considers a remote host anomalous once it contacts more than 15 local hosts within an unspecified time window, or when the remote host attempts a connection to a non-existing host. Kato et al. [13] proposed an IDS in which a remote host is labeled as a scanner if the number of TCP ACK/RST returned to the remote host within a time window is above a threshold. Some probabilistic models [15, 31] consider both the number of local hosts/ports accessed by a remote host, and how unusual these accesses are such that packets sent to rarely accessed IP address/port combinations have higher probability to exceed an anomaly score threshold where an alert is generated. This approach requires rich knowledge of the monitored network and dynamic updates of the hosts/ports probabilities.

A scan detection preprocessor plug-in called sfPortscan [23] in Snort [24] generates an alert when a remote host attempts to connect to more than a predefined threshold of local hosts (4 IP addresses is the default) or to more than a predefined threshold of ports (default 19) within a predefined time window (default 1 minute). However, by not exceeding the probing threshold within the specified time window, an adversary can easily evade detection. In an early version of the Bro IDS, a remote host is considered a scanner if it contacts more than a predefined threshold of local hosts or by contacting too many ports. Newer releases of Bro [1] come with a scan analyzer that considers many tunable parameters. TRW [11] is also implemented as a Bro policy such that scanners' traffic can be dropped by setting the appropriate

[5]Note that the reported false discovery rate (FD rate) in Table 4 is different than false positive rate (FP rate) as explained in Section 4.2.

interface between Bro and the corresponding router.

A hybrid approach that combines a variation of TRW and a credit-based connection rate limiting algorithm is proposed by Schechter et al. [25]. The new variation detects fast scanning worms that can generate thousands of connection attempts (to find vulnerable machines) before being caught if only TRW is deployed for scan detection. In addition to the drawback that limiting the rate at which first-contact connections can be initiated could block some legitimate hosts, this approach is unable to detect stealthy probes. A simplified variant of TRW that required less memory footprint and can detect vertical scanning is proposed by Weaver et al. [34]. A similar modification [19] considered vertical scanning and extended TRW to detect UDP and ICMP scans. The authors used a Bloom filter (a probabilistic space-efficient data structure for testing set membership) to filter the input to TRW so that only unique source and destination IP addresses, destination port and protocol are processed by TRW (which apparently eliminates the need for FC and SC in Algorithm 1).

Using a set of statistical features, Simon et al. [27] proposed a data mining classifier to detect scanners avoiding P2P and backscatter traffic. Their main rule in differentiating between scanning and P2P traffic relies on their observation that unlike scanners, the probability that remote hosts initiating P2P traffic make connection attempts to different local hosts on the same closed destination port is negligible. While P2P traffic that does not follow this rule might be mistakenly flagged as scanning activity, it is also easy for scanners to evade detection by contacting different ports at different local hosts so they do not exceed the thresholds.

Testing TRW on two 3G cellular network datasets, Falletta and Ricciato [5] observed that TRW marks remote hosts with P2P activity as scanners. The authors proposed an unproven heuristic metric to differentiate between scanning and P2P activity. LQS [2] is a lightweight real-time network scanning detection algorithm that, unlike TRW (and STRW), can detects both vertical and stealthy scanning (for discussion of LQS, see Section 2.3).

Generally, in addition to requiring a priori knowledge of the monitored network, it is hard to find the right tradeoff between detection (i.e., TP rate) and false positive rate in such threshold-based approaches. Moreover, the main dilemma is that lower false positive rate usually leads to a lower detection rate.

7 Concluding Remarks

Using failed connection attempts as an indication of network scanning activity remains a feature that scanners can not evade since their lack of knowledge of the accessible network hosts or services is what drives them to scan a network. The relatively subtle modifications to TRW introduced by STRW lead to dramatic reduction in false positives. In our empirical evaluation of TRW on network traces from two sites, across different configurations of parameters, between 19% and 50% of the scanners reported by TRW are false positives. While maintaining the same detection accuracy as TRW, STRW significantly reduces the false discovery rate (29% and 77% of TRW false positives are avoided in two datasets studied). We also provide guidelines on how to set appropriate values for the parameters of both TRW and STRW to enhance detection accuracy.

While using the EM technique as a proxy for approximated ground truth can only identify a subset of the scan detection false positive results, its high confidence level in identifying benign remote hosts makes it suitable as a preliminary evaluation method for scan detectors. We hope that by improving scan detection accuracy our work may encourage broader adoption of scan detection mechanisms and thus enable early defensive actions for mitigating network attacks that are preceded by such a reconnaissance phase.

Acknowledgments

The second author is Canada Research Chair in Authentication and Computer Security and acknowledges NSERC for funding the chair, and a Discovery Grant. Partial funding from NSERC ISSNet is also acknowledged.

References

[1] Bro intrusion detection system. Accessed: March 2009, http://bro-ids.org/.

[2] M. Alsaleh and P. C. van Oorschot. Network scan detection with LQS: A lightweight, quick and stateful algorithm. In *Proc. ASIACCS*, 2011.

[3] R. Basu, R. K. Cunningham, and S. E. Webster. Detecting low-profile probes and novel denial-of-service attacks. In *Proc. the IEEE Workshop on Information Assurance and Security*, 2001.

[4] C. Muelder, K.-L. Ma, and T. Bartoletti. Interactive visualization for network and port scan detection. In *Proc. RAID 2005*.

[5] V. Falletta and F. Ricciato. Detecting scanners: Empirical assessment on a 3G network. *International Journal of Network Security*, 9(2):143–155, September 2009.

[6] C. Gates. Coordinated scan detection. In *Proc. NDSS 2009*.

[7] C. Gates, J. J. McNutt, J. B. Kadane, and M. Kellner. Scan detection on very large networks using logistic regression modeling. In *Proc. the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, 2006.

[8] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. *Proc. IEEE Symposium on Security and Privacy*, 1990.

[9] J. Jung. Real-time detection of malicious network activity using stochastic models. *PhD thesis, MIT*, 2006.

[10] J. Jung, R. A. Milito, and V. Paxson. On the adaptive real-time detection of fast-propagating network worms. In *Proc. DIMVA*, 2007.

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

[12] T. Karagiannis, A. Broido, N. Brownlee, K. C. Claffy, and M. Faloutsos. Is p2p dying or just hiding? In *GLOBECOM Conference*, Nov 2004.

[13] N. Kato, H. Nitou, K. Ohta, G. Mansfield, and Y. Nemoto. A real-time intrusion detection system (IDS) for large scale networks and its evaluations. *IEICE Transactions on Communications*, E82-B(11):1817–1825, 1999.

[14] S. Lau. The spinning cube of potential doom. *Communications of the ACM*, 47(6):25–26, 2004.

[15] C. Leckie and R. Kotagiri. A probabilistic approach to detecting network scans. In *Proc. NOMS*, 2002.

[16] Z. Li, A. Goyal, and Y. Chen. Honeynet-based botnet scan traffic analysis. In *Botnet Detection*, pages 25–44, 2008.

[17] Z. Li, A. Goyal, Y. Chen, and V. Paxson. Automating analysis of large-scale botnet probing events. In *ASIACCS*, 2009.

[18] T. Moore and R. Clayton. Evil searching: Compromise and recompromise of internet hosts for phishing. In *Proc. FC*, 2009.

[19] V. Nagaonkar. Detecting stealthy scans and scanning patterns using threshold random walk. *Master's thesis, Dalhousie University, Canada*, 2008.

[20] NMAP. Accessed: Apr 2008, http://nmap.org.

[21] S. Panjwani, S. Tan, K. M. Jarrin, and M. Cukier. An experimental evaluation to determine if port scans are precursors to an attack. *Proc. Inter. Conference on Dependable Systems and Networks*, 2005.

[22] V. Paxson. Bro: A system for detecting network intruders in real-time. In *Proc. USENIX Security*, 1998.

[23] D. Roelker, M. Norton, and J. Hewlett. sfPortscan, September 2004. http://cvs.snort.org /viewcvs.cgi/snort/doc/README. sfportscan?rev=1.6.

[24] M. Roesch. Snort: lightweight intrusion detection for networks. In *Proc. LISA*, 1999.

[25] S. E. Schechter, J. Jung, and A. W. Berger. Fast detection of scanning worm infections. In *Proc. RAID*, 2004.

[26] U. Shankar and V. Paxson. Active mapping: Resisting NIDS evasion without altering traffic. In *Proc. IEEE Symposium on Security and Privacy*, 2003.

[27] G. J. Simon and H. Xiong. Scan detection: A data mining approach. In *Sixth SIAM International Conference on Data Mining*, 2006.

[28] A. Sridharan and T. Ye. Tracking port scanners on the IP backbone. In *Proc. LSAD*, 2007.

[29] S. Stafford and J. Li. Behavior-based worm detectors compared. In *Proc. RAID*, 2010.

[30] S. Staniford. Containment of scanning worms in enterprise networks. *Journal of Computer Security*, November 2003.

[31] S. Staniford, J. A. Hoagland, and J. M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[32] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, and K. Wu. Detecting distributed scans using high-performance query-driven visualization. In *Proc. the ACM/IEEE Conference on Supercomputing*, 2006.

[33] W. W. Streilein, R. K. Cunningham, and S. E. Webster. Improved detection of low-profile probe and DoS attacks. In *Proc. the Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, 2001.

[34] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms, revisited. *Malware Detection*, 27:113–145, 2007. Springer-Verlag Advances in Information Security.

[35] D. Whyte. Network scanning detection strategies for enterprise networks. *PhD thesis, Carleton University*, 2008.

[36] D. Whyte, P. C. van Oorschot, and E. Kranakis. Tracking darkports for network defense. ACSAC 2007.

[37] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: global characteristics and prevalence. *SIGMETRICS Perform. Eval. Rev.*, 31(1):138–147, 2003.