View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Springer - Publisher Connector

Harang and Mell *Security Informatics* (2015) 4:4
DOI 10.1186/s13388-015-0019-7

Security Informatics
a SpringerOpen Journal

RESEARCH Open Access

Evasion-resistant network scan detection

Richard E Harang[1*] and Peter Mell[2]

Abstract

Popular network scan detection algorithms operate through evaluating external sources for unusual connection patterns and traffic rates. Research has revealed evasive tactics that enable full circumvention of existing approaches (specifically the widely cited Threshold Random Walk algorithm). To prevent use of these circumvention techniques, we propose a novel approach to network scan detection that evaluates the behavior of internal network nodes, and combine it with other established techniques of scan detection. By itself, our algorithm is an efficient, protocol-agnostic, completely unsupervised method that requires no a priori knowledge of the network being defended beyond which hosts are internal and which hosts are external to the network, and is capable of detecting network scanning attempts regardless of the rate of the scan (working even with connectionless protocols). We demonstrate the effectiveness of our method on both live data from an enterprise-scale network and on simulated scan data, finding a false positive rate of just 0.000034% with respect to the number of inbound flows. When combined with both Threshold Random Walk and simple rate-limiting detection, we achieve an overall detection rate of 94.44%.

Keywords: Intrusion detection systems; Network scanning; Algorithms; Experimentation; Measurement; Security

Introduction

Network scanning – an attempt to enumerate and/or fingerprint hosts and services on some victim network – is a common precursor to an attack, and often has utility in post-incident forensics. As discussed in [1], precisely quantifying what a 'scan' is can be difficult. This is particularly true when one considers parallelized scans (e.g. from a botnet), extremely slow scans attempting to evade detection, and the potential for non-hostile scans. We focus on detecting 'horizontal' scans from a single source that attempt to discover the active hosts on a network (e.g., "Ping scans") or a particular service across a network (e.g., scans for SSH services on port 22). This approach may apply to detecting scanning botnets as each contributing source can be detected individually. We do not detect 'vertical' scans that attempt to enumerate the services on a single host.

A highly effective and widely cited method for detecting scanners is the Threshold Random Walk (TRW) algorithm [2]. TRW can often detect single source scanning after only 4 or 5 connection attempts. Related approaches include [3] and [4]. While effective, these approaches suffer from two limitations: they can be circumvented through mixing probe attempts with accesses to known active hosts, and they cannot handle scans using stateless protocols. Simpler thresholding methods, such as those in [6-9], count probe frequency within some time window and can be circumvented by an attacker slowing down the scan rate.

Thus, there is thus a need for a complementary scan detection method that 1) cannot be circumvented through knowledge of existing hosts, 2) does not require stateful connections or additional information about the internal structure of the defended network, and 3) cannot be evaded by simple rate-limiting approaches. In this paper, we develop such an approach and then show how it can be combined with TRW and simple rate limiting to form a highly effective ensemble approach.

Our method draws on a related insight to that developed with TRW in [1]: since scanners do not know the internal structure of the network they are more likely to access inactive nodes. However, where TRW examines the number of legitimate and illegitimate targets with which each source communicates, we examine the number of sources communicating with each target. This conceptual inversion of the high-level TRW logic enables us to avoid many limitations of TRW (particularly the ability to evade TRW by connecting to operational servers), while retaining TRW's advantages in terms of rate-insensitivity and rapid detection.

* Correspondence: richard.e.harang.civ@mail.mil
[1]U.S. Army Research Laboratory, Adelphi, MD, USA
Full list of author information is available at the end of the article

Springer

© 2015 Harang and Mell; licensee Springer. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

We proceed as follows. In section 2, we present related work and our differentiation from it. In section 3, we present details of our scan detection methodology. Section 4 analyzes the algorithmic execution and memory complexities. Section 5 describes our experimental data, and section 6 provides the empirical results. We conclude in section 7.

Related work

A wide variety of scan detection methods have been surveyed in [2]. Perhaps the most influential and significant of those is that of [1], in which the TRW method is proposed. This approach is based on the observation that in the course of enumerating an unknown network, a scanner will generate a relatively large number of unsuccessful connections; legitimate traffic by contrast should generate unsuccessful traffic only rarely. By examining a ratio of successful to unsuccessful connections, robust and rapid identification of potential scanners can be achieved. A weakness of the algorithm is that a scanner can boost its successful connection count by accessing known good servers, enabling additional probing of unknown addresses prior to detection. If enough known good servers are probed first (e.g., 6 or 7 for the published TRW parameters), TRW will permanently classify the scanner as benign and will ignore any subsequent scanning activity [3]. TRW can be modified to avoid such permanent labelling, but this causes a significant increase in the computation and memory requirements [3], while still leaving open the ability for an attacker to delay being categorized as malicious by seeding scan activity with accesses to known servers. Another weakness is that TRW only operates on stateful protocols like TCP that clearly define a "failed" connection. This limitation is due to TRW classifying stateless communication as scan activity in situations where there is a lack of bi-directional communication [3].

TRW has been used as a building block for other work. The work of [4] describes enhancing TRW with a severity metric to distinguish reconnaissance scanning from peer-to-peer scanning. A ranking metric for a TRW-based method is also applied in [5] to detect inter-domain scans. The work of [6] examines a different extension of the TRW method in which the target thresholds for the likelihood ratio are sequentially adapted according to some user-defined acceptable risk for a bounded sequence, thus allowing the TRW method to converge to a decision in (almost surely) finite time. The work of [7] modifies TRW to allow for scan detection in network backbones where there is asymmetric routing. It considers the ratio of distinct IPs contacted to distinct ports contacted within a time window, and labels IP addresses that show a pronounced asymmetry in either direction (i.e., an extremely high number of IPs contacted on just a few ports or an extremely large number of ports on a limited number of distinct IPs) as potential scanners.

The work of [8], similarly to TRW, uses a likelihood ratio test, however it focuses on comparing the empirical distribution of benign traffic with that of an assumed uniform distribution of scanning traffic. As noted in [1], an inaccurate empirical distribution may result in significant false positive results and an assumed uniform distribution of scanning traffic may present detection circumvention opportunities for attackers. Probabilistic methods are also used in [9] to construct a Bayesian belief network to detect anomalous packets that may be indicative of a scan and a correlation engine to attempt to collect these anomalous packets into sets that suggest scanning behavior. As in [8], the accuracy of this method depends in large part on the quality of the data used to build the statistical model that represents 'normal' traffic.

Other approaches include [10], in which the tools of social network analysis are applied to graphs constructed from netflow data to detect a wide range of intrusive behaviors, including scans. In [11], the RIPPER data mining tool is applied to a set of hand-crafted feature vectors in order to learn novel rules to classify network scans.

Simpler threshold based mechanisms (see, e.g., [12-15]), that simply tally connection attempts and alarm if the number of connections attempted by a single IP exceed some threshold within a certain period of time, have also been used. These can be effective for many types of scanning and were implemented in systems such as the Network Security Monitor [12] and Snort [13]. Later systems such as Bro [14] elaborated on this initial concept by also tracking failed connections in a side count for particular ports. Such techniques are lightweight and sufficiently effective on unsophisticated scanning activity that they continue to be widely used. More recent developments in this vein examine TCP flags attempting to find unusual sequences of flags, such as FIN flags being sent or received when no already established connection is present [15], or construct more elaborate feature vectors based on distinct IP counts of successful and unsuccessful connections [16].

Several methods address detecting groups of collaborating scanners, such as those coordinated through a botnet or other covert distributed system [17-20]. While our approach can detect individual scanners making up such a coordinated effort, we do not attempt to group detected scanners into collaborating groups.

Method

As stated previously, our method is designed to work with both stateful and stateless protocols in situations where knowledge of the currently active hosts is not available. It is also resistant to attackers deliberately accessing known

active servers in order to perform a TRW camouflage attack [3]. The method is illustrated in Figure 1. First, we examine inbound flows (flows from internal to external hosts), and score each internal host on the basis of the number of unique external hosts that have contacted it (to associate higher scores with 'riskier' behavior, we take the inverse of this number; see annotations for W in Figure 1). Next, each external host is scored on the basis of the highest score obtained by any communicating partner (see annotations for S in Figure 1). Finally, to reduce false positives, we track the number of distinct connections that each external host creates (see annotations for C in Figure 1). Combining the S and C measures allow us to classify each external host as a scanner or benign host.

Formally, we construct a directed bipartite graph $G = (V,E)$ with $V = L \cup R$, all external hosts in L and all internal hosts in R. We let the edge (x,y) indicate that there exists observed traffic from x to y, and add only edges of the form $\{(x,y) : x \in L,\ y \in R\}$ (so that only "inbound" traffic is recorded). We then assign scores W_y, S_x, and C_x as:

$$W_y = 1/\#|x \in L : (x,y) \in E|$$

$$S_x = \sup_y \{ W_y : (x,y) \in E \}$$

$$C_x = \#|y \in R : (x,y) \in E|$$

Where $\#|set|$ denotes the cardinality of *set*. The nodes with a high W are those that received communication attempts from a relatively low number of distinct hosts. The nodes with a high S are those that attempted to initiate connections to any such sparsely contacted hosts (the supremum of the edge connected W_y scores). And the nodes with a high C made many connections (without reference to the connectivity of the target).

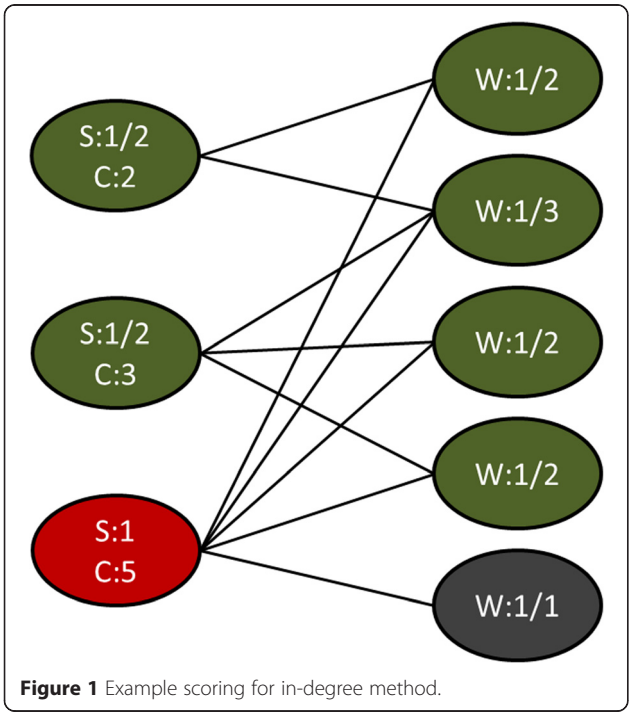

Figure 1 Example scoring for in-degree method.

Under the assumption that inactive hosts will receive far fewer inbound connection attempts than active hosts, and that scanners are disproportionately likely to attempt to both contact these inactive hosts and make a large number of connection attempts, it then follows that external nodes with high joint C and S scores are more likely to be scanners. If statistical information is known about the rate of incoming packets to active hosts, these scores may be used in a likelihood test similar to [1] or [8] to identify the probability that a particular external host is scanning. In the absence of this information, we may use unsupervised approaches such as thresholding or clustering to identify potential scanning nodes. A straightforward approach that does not require reliance upon strong distributional assumptions (e.g. normality of the distribution of contacts) is to choose some empirical quantile $\theta_s,\ \theta_c \in [0,1]$ and find thresholds α and β satisfying:

$$\alpha = \operatorname*{argmin}_i P(S_x \le i :\ x \in L) \ge \theta_s$$

$$\beta = \operatorname*{argmin}_t P(C_x \le t :\ x \in L) \ge \theta_c$$

where the probability measure P is obtained from the empirical distribution of S or C, as appropriate. We then designate an external host as a potential scanner if both $(S_x > \alpha)$ and $(C_x > \beta)$.

In Figure 1 we provide a simple worked example of the scoring method. We assume that the external IPs are the left partite component, and the internal IPs are the right partite component. The bottom-most external IP is a scanner, and the bottom-most internal IP is an inactive host. As the inactive host receives only a single connection from the scanner, it obtains a W score of 1; this is the largest across the W scores of the internal IPs connected to the scanner, so the scanner obtains an S score of 1. As the scanner contacts 5 internal IPs, it obtains a C score of 5.

Note that this method requires only directed flows from the L component to the R component in order to calculate the relevant scores. This feature allows the in-degree method to function in situations where asymmetric routing exists without any need for modification (c.f. [7], where the TRW approach required modification to function adequately under asymmetric routing conditions). We do not address this feature directly, but the experience of [7] suggests that direct deployment of the TRW method under asymmetric routing will be highly problematic.

Algorithm and execution complexity

This method can be implemented using a batch processing approach in time complexity O($m$log$m$) where m represents the number of flows but can achieve linear time with a slight variation. Most steps are O(m) while the calculations for α and β require O($m$log$m$) due to the requirement to sort two arrays containing S and C values respectively. If linear time complexity is needed, the calculations for α and β can be reused in subsequent batches provided that the time slices being evaluated are of the same length, or can be approximated through a number of on-line estimators [21]. The memory footprint required is at most O(m) provided one uses an in-place style sorting algorithm (e.g., in-place heapsort).

The algorithm is as follows:

1. Extract all flows whose source is in L and whose destination is in R in O(m) time.
2. Enumerate over all extracted flows to filter out those that do not connect distinct IP pairs. Use a hash table, LRHash, keyed by both IP addresses to determine if a flow connects distinct pairs. A single hash table lookup and insertion takes O(1). Since there are O(m) remaining flows, this operation takes O(m).
3. Enumerate over the set of flows with distinct IP pairs. For each flow, populate and update a hash table, RHash, by using the destination IP address as the key. For each processed flow, increment a count of source IPs communicating with the destination IP and update a variable containing the inverse of this value (the W metric). This uses O(m) time since there are O(m) flows and hash table lookups and updates are O(1).
4. Enumerate over the set of flows with distinct IP pairs. For each flow, populate and update a hash table, LHash, by using the source IP address as the key. For each processed flow, increment the count of destination IPs communicating with the source IP (this is the C metric) and append each destination IP to a list (one list for every hash key). This uses O(m) time since there are O(m) flows, hash table lookups and updates are O(1), and appending to a list is O(1).
5. Calculate the S metric for each source IP by accessing each key in LHash and traversing the local list to look up each destination IP in RHash. Assign S to be the largest W value found in the RHash lookups and store it in LHash. This takes O(m) to access each key, an amortized O(m) to traverse all destination IP lists, and O(1) for the RHash lookup. Overall this is O(m).
6. Calculate α in O($m$log$m$) by accessing each key in LHash, building a sorted array of the discovered S values, and extracting the value stored at the index ($\theta_S * length(list)$).
7. Calculate β in O(mlogm) by accessing each key in LHash, building a sorted array of discovered C values, and extracting the value stored at the index ($\theta_C * length(list)$).
8. Compare each source IP in LHash and its S and C values to α and β to identify the scanners in O(m) time.

This batch based algorithm can be executed with overlapping time windows since the linear execution option executes very quickly. The algorithm can then be run in small periodic increments (but still using large time windows), approximating an always-on online solution. A true online algorithm is more difficult to achieve because a single flow can change the S value in O(m) nodes in L. Even more problematic, a newly processed flow indicates that some amount of time has elapsed. Thus, previously process flows may need to drop out of the calculation for W, S, and C values for all nodes. Data structures to accommodate this are computationally expensive and so we suggest using linear complexity, continuously running, overlapping time window, batch jobs as a better solution.

Data

Our data sets consist of network flows, both live and simulated. Each flow contains source and destination IP addresses, port numbers, connection times, and TCP flags (used for TRW analysis).

Live data

Our live data was collected for 24 hours from a single network intrusion detection sensor. A total of 5,897,187 flows were observed, involving 454,510 distinct pairs of IP addresses. This data was known *a priori* to contain one port scan across a substantial portion of a class C subnet.

Due to the potentially sensitive nature of the live data, it has been anonymized in our results as follows:

1. Each distinct external class C subnet within the data has been replaced with a non-routable subnet selected uniformly at random from the set of non-routable subnets in 10.0.0.0/8.
2. Each distinct internal class C subnet within the data has been replaced with a non-routable subnet selected uniformly at random from 172.16.0.0/12.
3. Each final octet within each class C subnet (both internal and external) has been replaced with an octet selected uniformly at random from the range 1–255 (for the sake of simplicity we neglect the reduction in valid IP addresses caused by the assignment of gateway, network name, and broadcast addresses).
4. We do not report port information, other than noting whether source IP, destination IP, or both exhibit a fixed port during the reported scan.

Simulated data

To augment the live data, we simulate 3 scans by introducing appropriate flow records into the live data. We simulate an ICMP "ping" scan, a TCP SYN scan (with fixed source and destination ports), and a UDP scan (with fixed destination ports). Each simulated scan probes all 255 members of a class C subnet within the network perimeter from a spurious source IP address. Partial and rate-limited scans are simulated by subsampling some proportion of the full scans and distributing their event times uniformly over the duration of the flow being analyzed. Responses from active hosts to the simulated scanning packets were created as follows: ICMP packets generated no response; UDP packets generated either an ICMP Port Unreachable message, a mock UDP response packet, or no response (all with equal probability); TCP packets received an ACK for active hosts (followed by a final RST packet from the scanner). Inactive hosts created no response.

We note that our algorithm does not make use of subnetting information except at the presentation layer, and hence a very sparse scan across a class B subnet (for example, a single random IP within each class C subnet contained within the class B subnet) would yield the same results, however this renders the display of results less compact, and so is omitted for space.

Results

We first used our in-degree algorithm as a guide to discover novel scans within the live network data, as well as the scan known *a priori*, while examining the impact of various parameter settings. We then explore an ensemble scan detection solution by combining our algorithm with TRW and rate-based scanning. We show how TRW complements our approach by both approaches detecting different sets of scanning IPs. We furthermore show how pre-filtering our in-degree method with a rate-based scanning method can substantially reduce the false positive rate. We then compare our findings against a manual analysis of the same data. For our last data set, we investigate in-degree and TRW detection of the simulated scans injected into the live data. We conclude this section with an analysis of the difficultly of an attacker simultaneously circumventing both our in-degree approach and TRW.

In-degree method results on live network traffic

After the initial processing of live data as described above, 509 external hosts were identified as initiating connections with nodes within the portion of the network observed by the sensor. The S and C values were tabulated as described in section 3 and are presented in Figure 2 and Figure 3. Total counts that exceed joint thresholds of C and S are given in Table 1.

Marginal distributions of S and C suggest that the bulk of the data (89.2%) has an S score of 0.333 or less, and 83.3% of the data has a C score of 1 or less. The joint distribution indicates that there is a slight dependence between S and C, with approximately 2.0% (10/509) of the data having both $S > 0.333$ and $C > 1$ versus 1.8% if the scores were independent. The data having these values are highlighted in bold in Table 1 (bold italics indicates the presence of scans that were not detected by any combination of methods examined, as discussed below). At the other extreme, approximately 86.4% of our joint data falls into the lower bins in both dimensions (versus an expected value of 74.3% assuming independence). A permutation test yields a one-tailed p-value of 0.0178, indicating slight but significant correlation.

Detailed examination of the 10 external IPs in the bold cells in Table 1 indicated that 2 were false positives. One was determined to be the result of a web-based advertising network rotating through a pool of IP addresses serving auto-refreshing content to 4 otherwise idle hosts. The second was the result of network time protocol (NTP) traffic delivered to 2 otherwise inactive hosts. There was thus a .000034% false positive rate with respect to the number of original flows and .00044% with respect to distinct pairs of IP addresses. By either measure, the false positive rate is extremely low.

The remaining 8 were clearly identifiable as scan attempts, and confirmed to be such after examination by network analysts. Summaries of two of the most obvious scans, consisting of two external IP addresses directed traffic to 84 and 261 distinct IP addresses, respectively, are presented in Table 2 (scans 1 and 2) as well as a more subtle one (scan 3) discussed below. The TCP scan (scan 1) from Table 2 was known *a priori* to be in the data, and was successfully detected by our algorithm. While it is gratifying to note that these two scans were detected, it should be noted that these were rather obvious scans readily detected by even simple windowing methods. Scan 1 in Table 2 was perhaps somewhat rate limited, scanning just over 4 IP addresses per second; however, scan 2 in Table 2 made no attempts at subtlety, probing substantial portions of two class C subnets in less than 3 seconds.

Of perhaps greater interest are those scans that did attempt some degree of subtlety, presumably to evade traditional rate-based scan detection methods. Scan 3 in Table 2 shows a summary of one such scan detected in the live data by our method, in which just 7 IP addresses were scanned over the course of approximately 7.75 hours. Consultation with network analysts suggests that there is an extremely high likelihood that this traffic represents a deliberate attempt to evade rate-based portscan detection algorithms. Note that the ICMP protocol is not one that is supported by most stateful connection-based scanners [3],

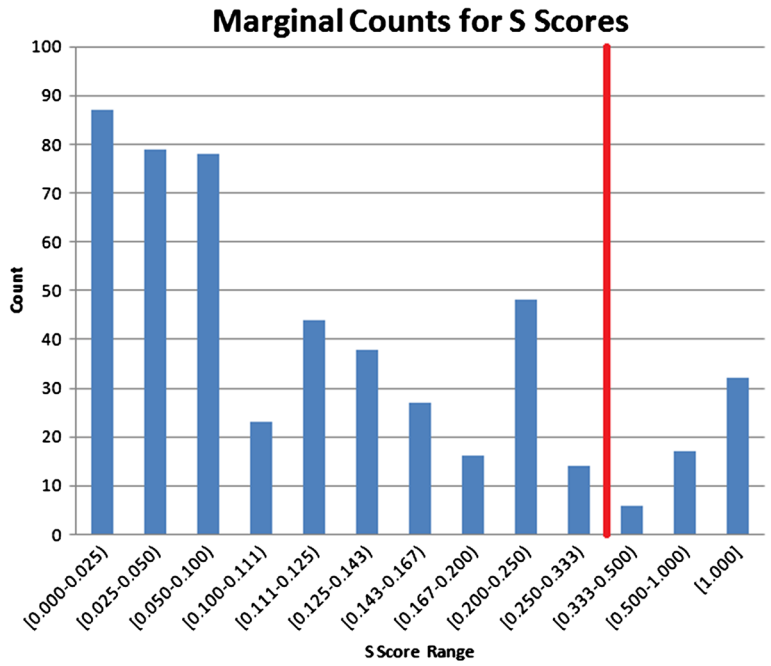

Figure 2 Marginal S Counts for scores; the red line indicates the selected cutoff.

and hence this scan would not have been detected by those methods.

Our results for the in-degree method alone are summarized in Table 3. This includes results for a 'filtered' version of in-degree where we use a rate-based detection scheme to filter out obvious scans (explained in more detail in section 6.2.2). We define a lateral scan as a scan in which a single target port was identical over all internal IP addresses scanned. We define a network enumeration scan as one in which either a portless protocol (i.e. ICMP) was used, or a single destination port per IP was contacted that varied from IP to IP in what appears

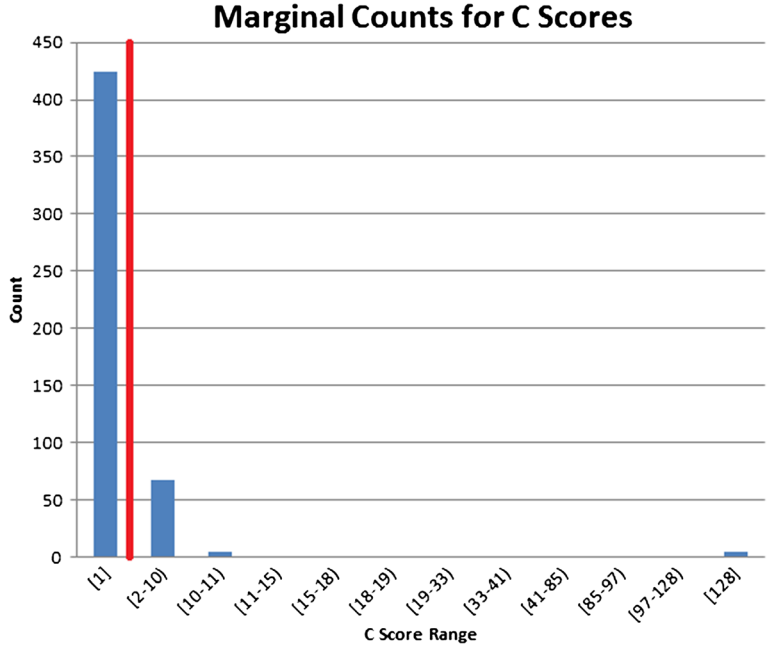

Figure 3 Marginal C Counts for scores; the red line indicates the selected cutoff.

Table 1 Cumulative counts greater than or equal to joint values of C and S

	C score						
S score	***1***	***2***	***10***	***18***	***85***	***97***	***128***
0.010	*509*	*85*	*17*	*10*	*6*	*5*	*4*
0.050	*343*	54	17	10	6	5	4
0.200	*117*	40	15	9	6	5	4
0.250	*69*	17	***2***	***2***	***1***	***0***	***0***
0.333	*55*	**10**	**2**	**2**	**1**	**0**	**0**
0.500	*49*	***7***	**2**	**2**	**1**	**0**	**0**
1.000	*32*	***7***	**2**	**2**	**1**	**0**	**0**

Bold font indicates that the S and C values were greater than or equal to the data-determined thresholds of 0.333 and 1.0, respectively; bold italics indicates the presence of scans at those S/C values that were not detected (see text for details). Note that these represent *cumulative* counts, and so, for example, the value of 10 for $S \geq 0.333$ and $C \geq 1$ includes all values below and to the right.

to be a largely random manner. We define a service enumeration scan as one where multiple connections were attempted to each internal IP across a fixed set of ports, with substantially the same set attempted for each such IP. We do not include false positives in the table but restate that 2 inactive hosts received legitimate traffic that resulted in a false positive result for both filtered and unfiltered methods.

Composition of methods on live data

We now explore how the in-degree method complements TRW detection and how rate-based detection methods can reduce in-degree false negatives.

TRW and in-degree method

The TRW algorithm was applied to the same data for comparison and the results are shown in Table 4. The data shows that the in-degree and TRW algorithms are strongly complementary, with very little overlap between the scans detected by the two methods. This suggests that our method successfully detects many scans that the TRW algorithm cannot (and vice versa).

Pre-filtering in-degree data with rate-based scanning

We next applied a simple rate-based detection scheme as a pre-filter to the in-degree algorithm (filtering out flows corresponding to detected scans) to obtain the results in Table 5. For this, we flagged an IP as a scanner if it completed more than 5 connections to distinct IP/port pairs within any 5-second window. Removing these scans before applying the in-degree method significantly enhanced the ability of the in-degree method to detect the remaining scanners; it did not improve the detection rate of the TRW algorithm. As all three scans removed by rate-limiting were UDP scans, and the TRW algorithm is not capable of detecting UDP scans in the absence of a network state oracle of some form, the removal of these scans did not alter the information available to the TRW algorithm or its performance. Three scans were detected and filtered in this way. These three scanned large portions of the network, increasing the number of connections to internal IPs, and thus lowered their W scores. In 6 cases these obvious scans lowered the S scores of other scanners (for the unfiltered in-degree instance), interfering with the ability of the in-degree method to detect them. Service enumeration scans (as defined above) were particularly vulnerable to this problem.

Table 5 then represents our most important result. Of the 36 known scans in our dataset, the tri-algorithm approach detected 34 of them. 3 were detected solely by the rate limiting approach. 13 were detected solely by TRW. 12 were detected solely by the in-degree algorithm. 6 were detected by both TRW and the in-degree algorithm. The overall detection rate for the tri-algorithm approach was then 94.44%. This compares to individual detection rates of just 50.00% for in-degree,

Table 2 Scan detected from live traffic

Field	Value(s) – scan 1	Value(s) – scan 2	Value(s) – scan 3
Scan duration	6 minutes, 7 seconds	2.8 seconds	7 hours, 48 minutes
Protocol	TCP	ICMP and TCP	ICMP
Flags	SYN or SYN-ACK	SYN or SYN-ACK (TCP only)	N/A
Source IP (anonymized)	10.141.12.103	10.97.54.7	10.60.88.39
Source ports	Constant	Varied between 4 values	N/A
Destination IPs (anonymized)	84 within 172.40.102.0/24	29 within 172.198.57.0/24	7 within 172.198.57.0/24
		110 within 172.45.99.0/24	
		122 within 172.110.117.0/24	
Destination ports	Constant	Constant (when TCP)	N/A
Packets transmitted	3 per contact	6 (ICMP) or 1 (TCP) per contact	1 or 3 per contact
Total bytes transmitted	186 per contact	420 (ICMP) or 62 (TCP) per contact	70 bytes per packet
Data bytes transmitted	24 per contact	48 (ICMP) or 8 (TCP) per contact	8 bytes per packet

Table 3 Detection counts by class of scan

Type	Detected (unfiltered)	Detected (filtered)	Not detected
Lateral port (TCP)	6	9	0
Lateral port (UDP)	0	0	3
Network enumeration (any)	2	5	0
Service enumeration	0	4	2
Totals	8	18	5

Table 5 Cross-tabulation of TRW and filtered in-degree detections

	TRW		
In-degree (pre-filtered)	**Detected**	**Not detected**	**Total**
Detected	6	12	18
Not detected	13	5*	18
Total	19	17	36

*3 of these 5 were detected by the rate-limiting scan detection algorithm and removed.

52.78% for TRW, and 8.33% for rate-limiting. The dramatic increase in the overall detection rate highlights the complementary nature of these three algorithms.

Manual analysis

We performed a manual analysis of the inbound traffic subset of the 5.9 million flow records identified above, using a variety of internal analysis and aggregation tools, as well as direct examination of the flows. This required significant analyst effort and thus is a completely infeasible method for routine scan detection. Our analysis revealed the 2 scans not detected by any automated detection method, both contained within the darker shaded cells in Table 1. These two appeared to be service enumeration scans, resulting in higher S scores for each scanned host as multiple services were requested. Note that as this was a manual analysis, we cannot rule out the possibility of additional false negatives. Reducing the threshold for S to 0.250 would have successfully detected both such scans at the cost of an additional 5 false positives (we also note that proposed modification to the TRW described in [3] would have enabled the TRW method to detect them). 25 external hosts also delivered a single contact to nodes which had not received any other traffic. While these cannot be ruled out as extremely slow scan attempts, they were determined to be most likely the result of misdirected traffic.

Simulated data

Simulated data was included in the raw data as described above in an attempt to precisely quantify the effect of scan size on the likelihood of detection. Values for α and β were selected as described above, using values of 0.75 for θ in both cases. Scans were directed at a single moderately populated subnet with 119 active IP addresses out of the space of 254 possible ones (excluding standard gateway and broadcast addresses). For each trial, the indicated number of probes were directed to random addresses in the subnet, and the in-degree algorithm run without further refinement. In all cases, the cutoff for C was 2 (preventing detection of scans consisting of a single probe) and the cutoff for S was 0.333. Scans were reliably detected with as few as 4 IPs scanned per attempt, and in no case did a single scan of 5 or more scanned IPs escape detection. Despite the creation of additional connections that could potentially have reduced the S score of some hosts as described above, our inclusion of simulated scans did not alter the detection rates for any of the scans detected in our initial analysis; we therefore only report the detection results for the additional, simulated scans. Results are shown in Figure 4.

Table 4 Cross-tabulation of TRW and in-degree detections

	TRW		
In-degree	**Detected**	**Not detected**	**Total**
Detected	2	6	8
Not detected	17	11	28
Total	19	17	36

As successful connections are only defined for TCP connections, the TRW method was applied only to the simulated TCP scans. Results are given in Figure 4. While the TRW method did successfully identify the majority of simulated scans as the number of probes per IP address approached 10, the success rate was markedly less than that of the in-degree method. The TRW results for probes 1–5 were expected as the construction of TRW requires a sufficient history of failed probes in order to be able to alert on a scan. The deficiency of TRW when compared to in-degree for probes 6–10 reflects the fact that probes to active services (prevalent in this subnet) can camouflage scanning activity against TRW detection. Conversely, in the operational data in section 6.2, scans generally involved a large number of probes per scan, resulting in a much higher TRW performance.

Circumvention analysis

While the in-degree method shows excellent performance when detecting random scans on the basis of low in-degree count, our false negative and rate-limited scan results point out a potential evasion technique: generating multiple connections to each host from distinct IP addresses in order to artificially decrease the S scores. Preliminary work has shown that under modest assumptions, it is not difficult to construct a 'chaff' set of connections from a set of scanning IP addresses that will

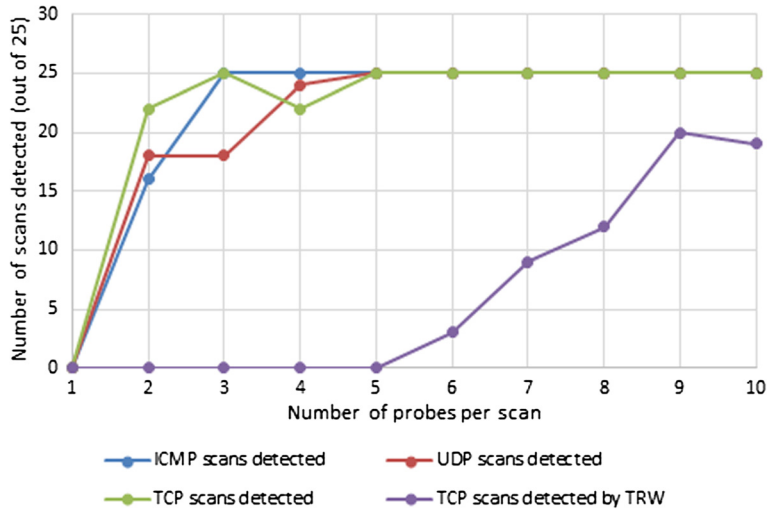

Figure 4 In-degree and TRW detection of simulated scans.

reduce the *S* scores sufficiently to avoid detection under this method, while simultaneously maintaining a low enough rate to evade rate-based intrusion detection systems. As shown above, however, the TRW method of [1] forms an excellent complementary method to ours (for TCP based scans) such that it will be difficult for an adversary to avoid both detectors simultaneously. The TRW method and its variants can be evaded by maintaining a sufficiently high ratio of successful to unsuccessful connections [1,3] while our method ultimately forces a potential attacker to create numerous unsuccessful connections to evade detection. In order to simultaneously maintain both a non-alerting TRW score and a non-alerting *S* score, the total number of connections that the attacker must therefore create per inactive IP address probed to avoid detection increases significantly. However, note that there are no limits on the rate at which any feasible covert scanning may be accomplished. By reintroducing rate-based detection methods, which limit the total rate of connections regardless of status, the combination of these three methods places limits on the total rate at which new IPs can be scanned without detection. In future work, we will investigate the synthesis of the three methods and theoretical limits on the potential for covert attacker scanning rates.

Conclusions

We have presented a novel approach to scan detection that operates in a complementary fashion to the Threshold Random Walk (TRW) method of [1]. Instead of just providing another scan detection methodology, our approach detects previously unmitigated TRW circumvention activity as well as activity that operates using connectionless protocols. Our novel method has a low time complexity, and has been applied to real-world network data both alone and in conjunction with the TRW and rate-limiting scan detection methods. The in-degree method is shown to detect scans not identified by the TRW method, and to do so with increased accuracy when the data is pre-processed to remove scans detected by a simple rate-limiting method. Further application to randomly generated scans have shown that – under the assumption that target IPs are picked at random from the available addresses in the subnet, and the subnet in question is not completely allocated – the in-degree method is extremely effective at picking out novel scans. While doing this, the in-degree method maintains a very low false positive rate.

We combined in-degree method with TRW and simple rate limiting to form a highly effective ensemble algorithm. Since TRW and in-degree work using distinct data sets (internal node behavior versus external node behavior), the combination makes it extremely difficult for scanners to operate without detection and mitigates known circumvention approaches. Deliberately connecting to know active hosts prior to probing unknown addresses to circumvent TRW is likely to be detected by in-degree and 'chaff' connections designed to circumvent in-degree are likely to be detected by TRW.

Abbreviations

ICMP: Internet Control Message Protocol; IP: Internet Protocol; SSH: Secure Shell; TCP: Transmission Control Protocol; TRW: Threshold Random Walk; UDP: User Datagram Protocol.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

PM and RH jointly conceived the initial version of the in-degree algorithm. PM developed an efficient implementation of the in-degree method in Python. RH performed the experimentation. Both authors contributed equally

to writing and data analysis. Both authors read and approved the final manuscript.

Acknowledgments
This research was sponsored by the U.S. Army Research Labs and the National Institute of Standards and Technology (NIST). It 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, NIST, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation hereon. The authors would like to thank Harold Booth of NIST for assisting with the initial formulation of the in-degree concept.

Author details
[1]U.S. Army Research Laboratory, Adelphi, MD, USA. [2]National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

Received: 26 January 2015 Accepted: 23 April 2015
Published online: 09 May 2015

References

1. J Jung, V Paxson, AW Berger, H Balakrishnan, *"Fast portscan detection using sequential hypothesis testing,"*. IEEE Symposium on Security and Privacy, 2004, pp. 211–225
2. M Bhuyan, DK Bhattacharyya, JK Kalita, "Surveying Port Scans and Their Detection Methodologies". Comput J **54**(10), 1565–1581 (2011)
3. P Mell, R Harang, "Limitations to Threshold Random Walk and Mitigating Enhancements,", in *First IEEE Conference on Communications and Network Security (submitted)*, 2013
4. V Falletta, F Ricciato, "Detecting scanners: empirical assessment on a 3G network". Int J Network Secur **9**(2), 143–155 (2009)
5. L. Aniello, G. Lodi and R. Baldoni, "Inter-domain stealthy port scan detection through complex event processing.," Proceedings of the 13th European Workshop on Dependable Computing. (ACM, Pisa Italy, 2011)
6. X Chen, *"New Sequential Methods for Detecting Portscanners," arXiv preprint, 1204(1935)*, 2012
7. A Sridharan, T Ye, S Bhattacharyya, *Connectionless port scan detection on the backbone*. Performance, Computing, and Communications Conference, 2006
8. C Leckie, R Kotagiri, *A probabilistic approach to detecting network scans*. Network Operations and Management Symposium, 2002, pp. 359–372
9. S Staniford, JA Hoagland, JM McAlerney, Practical automated detection of stealthy portscans. J Comput Secur **10**(1/2), 105–136 (2002)
10. Q Ding, N Katenka, P Barford, E Kolaczyk, M Crovella, *Intrusion as (anti) social communication: characterization and detection*. Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 886–894
11. GJ Simon, H Xiong, E Eilertson, V Kumar, *Scan detection: A data mining approach*. Proceedings of the Sixth SIAM International Conference on Data Mining, 2006, pp. 118–129
12. LT Heberlein, GV Dias, KN Levitt, B Mukherjee, J Wood, D Wolber, *A network security monitor*. IEEE Computer Society Symposium on Research in Security and Privacy, 1990, pp. 296–304
13. M Roesch, *Snort – lightweight intrusion detection for networks*. Proceedings of the 13th USENIX conference on System administration, 1999, pp. 229–238
14. V Paxson, Bro: a system for detecting network intruders in real-time. Comput Netw **31**(23), 2435–2463 (1999)
15. M Dabbagh, AJ Ghandour, K Fawaz, WE Hajj, H Hajj, *Slow port scanning detection*. 7th International Conference on Information Assurance and Security, 2011, pp. 228–233
16. M Alsaleh, PCV Oorschot, *"Network scan detection with LQS: a lightweight, quick and stateful algorithm"*. Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, 2011
17. C Gates, *"Coordinated scan detection"*. Proceedings of the 16th Annual Network and Distributed System Security Symposium, 2009
18. R Baldoni, GAD Luna, L Querzoni, *"Collaborative Detection of Coordinated Port Scans"*. Midlab Technical Report, 2012
19. M Kang, J Caballero, D Song, *"Distributed evasive scan techniques and countermeasures,"*. Detection of Intrusions and Malware, and Vulnerability Assessment, 2007, pp. 157–174
20. Y Zhang, B Bhargava, Allocation Schemes, Architectures, and Policies for Collaborative Port Scanning Attacks. J Emerg Technol Web Intell **3**(2), 154–167 (2011)
21. M Greenwald, S Khanna, Space-efficient online computation of quantile summaries. ACM SIGMOD Record **30**(2), 58–66 (2001)

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Immediate publication on acceptance
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com