A Modified Multi-Resolution Approach for Port Scan Detection

Hwashin Moon, Sungwon Yi, and Keeseong Cho
Electronics and Telecommunications Research Institute
Broadcasting and Telecommunications Convergence Research Lab.
Daejeon, Korea
{hsmoon, sungyi, chokis}@etri.re.kr

Abstract—Although port scan detection techniques have been widely adopted by the modern network based security systems, the effectiveness of these techniques can significantly be limited since the detection performance heavily relies on the statically determined detection threshold. To tackle the problem, a multi-resolution approach called MRDS, maintaining multiple monitoring windows with the corresponding detection thresholds, has been proposed. However, deploying such technique in a high speed network is not easy due to the time and space complexity required for calculating the number of unique destination addresses contacted in the multiple monitoring windows. In this paper, we present a novel failed flow dispersion estimation technique, called Multi-Window State Map (MWSM), which requires a small amount of memory and a constant number of memory access for implementing the multi-resolution concept. We then extend the proposed MWSM into a complete port scan detector. Simulation results with real world traffic traces indicate that the proposed estimation technique manages the expected relative error and average standard error of less than 0.8% and 9% respectively and thus the MWSM based detection scheme reduces false positives by 60% compared to MRDS.

I. INTRODUCTION

For the last decade, port scans have been extensively used by attackers for finding vulnerable hosts. On performing port scans, an attacker sends probing packets to map the target network including types of protocols, services supported by the target, and the active IP addresses as well. In general, the majority of probes result in negative or no responses due to the lack of prior knowledge on the target network, and the traffic caused by such activities exhibits unusual properties such as frequent failed connection attempts compared to the legitimate one [1][2][3]. Port scan detection techniques leverage such traffic anomalies as the evidence of malicious activities and effectively identity attackers. Therefore, port scan detection has been widely adopted by the modern network based security systems such as network based intrusion detection systems (NIDS) and firewalls [4][5].

Although there are several features used to characterize and quantify traffic anomalies, the number of unique destination addresses contacted (destination address dispersion) [6][7][8] and the number of unique failed connection attempts (failed flow dispersion) [9][10] are the two most popular features in the NIDS literature. Regardless of the features and detailed scan algorithms used in port scan detection, these techniques rely on a simple assumption: Given a feature and a time window, there exists a single threshold that can distinguish port scanners from normal traffic. Under the assumption, the detection performance and sensitivity heavily rely on the choice of statically configured threshold value.

This research was supported in part by KCC and MKE grants 2009-S-026-01.

To address the problem, V. Sekar et al. [11] proposed a new scan detection and containment scheme called Multi-Resolution Detection System (MRDS). Instead of using a single detection threshold, MRDS maintains multiple monitoring windows with the corresponding detection thresholds. In MRDS, a smaller time window is used to detect high scan rate attacks and a larger one is used for slow scan attacks.

Although it is believed that the multi-resolution based approach helps in reducing false alarms and detects a wide spectrum of scan attacks, a detailed performance study showed the following limitations. First, managing destination addresses for each host in a high speed network is not trivial. In [11], authors claimed that destination address dispersion exhibits a concave growth trend (as a function of the time window size) in most cases, suggesting the multi-resolution approach. However, our traffic analysis showed the concave trend is significantly weakened for heavy traffic users (roughly top 1%). In addition, such hosts constantly changed over time. As a result, the heavy users can easily exhaust the memory of MRDS and significantly degrade the performance in calculating destination address dispersion.

Second, a greedy technique exploiting multiple parallel connections has been widely deployed in peer-to-peer (P2P) applications [12][13]. For example, a p2p node can issue hundreds of query messages to peers in order to shorten the time required for locating a target file, and launch multiple connections for downloading a single file. With the proliferation of the P2P applications, the accurate identification of attackers from such aggressive normal traffic is becoming increasingly difficult.

To tackle these problems, we developed a scan detection technique based on the multi-resolution concept. Instead of using destination address dispersion, the proposed technique called Multi-Window State Map (MWSM) calculates failed flow dispersion over a small hash table for each internal host. The hash table based approach of MWSM helps in not only providing a constant processing time but also making memory consumption predictable. In addition, failed flow dispersion helps in identifying attackers from legitimate but still aggressive traffic users, and thus significantly reduces false positives. Trace driven simulation results indicate that the proposed MWSM scheme accurately estimates failed flow dispersion. In fact, MWSM limited the expected relative error of less than 0.8% with 0.94M bytes of memory in our experiments. When extended to a complete scan detector, MWSM reduced the number of false positives by 60% compared to MRDS.

The remainder of this paper is organized as follows: We summarize and discuss the MRDS scheme in Section II since it is used as the base model of the proposed scheme. The proposed
The multi-resolution detection system (MRDS) is based on the following observations: while the connection patterns of normal users are bursty at short timescales, the average connection rate becomes very low for longer timescales. In addition, the connection patterns of normal users exhibit a significant amount of locality. Under such traffic characteristics, scan detectors rely only on a single threshold and often fail to detect attacks or result in many false alarms. To effectively detect a wide range of attacks, MRDS proposed to use multiple monitoring windows with different threshold values.

Here, finding proper thresholds is still important since the basic detection mechanism of MRDS follows the single threshold framework for a given window. By casting the problem in a cost optimization framework, MRDS obtains the optimal assignments of <window size, target scan rate> pairs in a sense of minimizing the security cost, which is a function of false positives and false negatives. For each internal host, MRDS maintains a set of destination addresses contacted (destination address dispersion) for each window size and raises alarms if the monitored destination address dispersion exceeds the predefined value, which is the window size times the target scan rate. Considering the fact that the sophisticated attacks such as a varying scan rate attack continue to be reported in the wild [14], the benefits that multi-resolution approach can bring are evident.

To examine the capability of MRDS, we performed extensive simulations using trace files captured from a class B campus network 1. Figure 1 (a) shows the average destination address dispersion of 5218 internal hosts under different window sizes. We examined the traffic packet by packet and excluded flows that were identified as attacks to focus on the behavior of normal hosts. While most of hosts (99%) exhibit a concave growth trend in destination address dispersion and thus it is converged to 12, the top 1 percent hosts of heavy users show quiet different connection patterns 2. The destination address dispersion of the heavy users continues to grow until the window size reaches a maximum value and the growth trend is no longer concave. Therefore, to avoid memory exhaustion by these heavy users and resulting performance degradation, the memory consumption in MRDS should be carefully controlled. In addition, heavy users constantly changed over time. Among 142 hosts of top 1% from three different traces of one site, we found only 2 hosts in common. It implies that a static IP filtering based approach such as whitelist may not be a remedy for the problem.

In the next experiments, we compared the effectiveness of destination address dispersion and failed flow dispersion in identifying attacks since they are the two most widely used traffic features in the NIDS community. Figure 1 (b) shows the destination address dispersion and failed flow dispersion of 5220 internal hosts over the longest monitoring window (500 seconds). In the Figure, two attacks are marked with red stars. The attack marked in the upper right corner is a botnet scanning port 6129 and the identification of the other one is unknown. It scanned random ports and random addresses with one scan per second on average. Assuming that the goal of a scan detector is to detect all instances of attacks, possible thresholds for destination address dispersion and failed flow dispersion are plotted as vertical and horizontal lines, respectively. While the detection relying on destination address dispersion caused 46 false alarms on our three traces with zero false negative, it was reduced to 2 when failed flow dispersion was used.

In summary, we found the following limitations through the traffic analysis and simulations. First, the multi-resolution approach can result in memory exhaustion and performance degradation due to heavy traffic users unless the memory usage is carefully controlled. Second, destination address dispersion is less effective in identifying attackers than failed flow dispersion. However, it is generally believed that counting failed flow dispersion is more difficult than counting destination address dispersion in high speed networks. In the next section, we describe how the proposed scheme solves these problems.

III. THE PROPOSED FAILED FLOW DISPERSION ESTIMATION SCHEME

We now present a novel failed flow dispersion estimation architecture, called Multi-Window State Map (MWSM). We first explain a destination address dispersion estimation mechanism within the multi-resolution concept and describe how it is extended to the proposed failed flow dispersion estimation scheme (MWSM).

A. Estimation of Destination Address Dispersion

A simple way for counting destination address dispersion on multiple monitoring windows is (1) binning outgoing traffic into the \( w \) second non-overlapping intervals (time bins) and (2) managing the set of destination addresses in each time bin as in [11]. In this case, the destination address dispersion of a host for a window size of \( iw \) seconds can be obtained by calculating the cardinality of the union of the \( i \) sets of destination addresses managed by the last \( i \) consecutive time bins. However, the exhaustive memory accesses in calculating the union operation can be a major obstacle in deploying a multi-resolution based scan detector in high speed networks.

---

1The details on the trace files and the experimental environment are described in Section IV.
2Most of heavy users are running on the applications such as email, auth, P2P, and web, and hosts with P2P applications account for 58% of the heavy users.
The number of memory accesses in the union operation can significantly be reduced if all the sets are mutually disjoint since the cardinality of union of sets can be calculated by the simple summation of cardinality of each set. With the multi-resolution framework, one way to manage all the sets to be mutually disjoint is to allow an entry (flow) only in the latest time bin. The entry can be counted comprehensively within a larger monitor window, which includes the latest one. However, finding the latest time bin still requires search operations in the sets for the each time bin and thus makes it difficult to provide the constant memory access time.

To address the problem, we propose a MWSM scheme implementing the disjoint set property on the bitmap scheme used in flow counting in [15] [16] [17]. To implement the disjoint set condition, we introduce a concept called virtual time stamp (VTS) and use a hash table to store VTS instead of a bit in the bitmap scheme. Here, VTS is a time identifier representing a time bin to make the union operation simple. Therefore, it is increased by 1 after each \( w \) seconds, and it is set to 1 if it reaches the maximum time bin size \( (N) \).

Figure 2 illustrates how VTS and hash table work in the MWSM scheme. To estimate destination address dispersion, each internal host manages a hash table and a vector \( q \) representing the number of hash lines occupied by each VTS as \( q = (q_1, q_2, ..., q_N) \), where \( q_i = |S_i| \) and \( S_i \) is the set of destination addresses monitored over the \( i \)th time bin. When a packet arrives, it is led to the hash table assigned to the source address, and the destination address, destination port \( \text{destination address, destination port} \) is hashed to find the corresponding index within the hash table. The current VTS \( vts_i \) is compared with the VTS \( vts_{s_b} \) in the hash table. If the hash table is empty (case 1 in Figure 2), the hash line is updated with current VTS, and \( q_i \) is incremented by one. If the current VTS is equal to the VTS in the hash table (case 2 in Figure 2), \( q \) remains the same. If two VTSs are different (case 3 in Figure 2), the hash line is updated with the current VTS, and the \( q_i \) is decremented by one and \( q_{i+1} \) is incremented by one. The sequence of operations for the three cases is shown by the solid, dotted, and dashed line in Figure 2. At the end of each time bin \( (w) \) seconds, we get the estimated destination address dispersion from \( q \) using the bitmap technique described in [16].

While the hash table with a small VTS size significantly reduces the number of memory accesses, the flow management based on hash tables has a known problem, time-out. In general, a flow counting scheme based on hash tables requires a time-out mechanism to purge the entries which occupy hash lines longer than the maximum monitoring window \( (Nw) \) seconds. Without such a mechanism, the estimation will continue to grow and result in over-estimation. Neither the sequential table checking nor the entire table reset can be an effective solution since they involve either the huge time delay or the loss of information. In particular, the loss of information can be more critical in security systems since it implies a security hole and can be exploited by malicious attackers.

We solve this problem by extending the single hash to dual-hash architecture. The proposed dual-hash table architecture consists of two identical hash tables. Each hash table records all the set membership for \( Nw \) seconds alternately, and the entire hash table is cleared before it is used. Therefore, MWSM can always keep the record of previous \( Nw \) seconds in one table \( (T_1) \) while the current set membership is recorded in the other hash table \( (T_2) \). Although only one hash table is used to record the set membership, both tables are used in computing the flow dispersion. For example, assume that one hash table \( (T_1) \) is used for the last \( i \) time bins. The destination address dispersion for \( Nw \) seconds can be computed by merging the data of two tables: We use the data of \( iw \) seconds from \( T_1 \) and the data of the most recent \( (N - i)w \) seconds from \( T_2 \).

Despite the increased memory consumption, we believe that the dual hash architecture is an effective solution, considering the following benefits it can bring. First, the flow expiration mechanism does not involve any data loss or additional delay. Second, due to the small VTS size and hash size (the number of entries in hash table), the total memory consumption is still very small. Third, the dual hash architecture plays a key role in estimating failed flow dispersion and is described in the following subsection.

B. Estimation of Failed Flow Dispersion

To detect a failed connection attempt, a detector has to either 1) rely on the explicit response from the receiving host/router (TCP RST or ICMP unreachable message) or 2) periodically check whether the connection request is granted. The former is easier to implement since the detector has only to monitor a handful of response packets. However, its performance heavily relies on network-wide cooperation. For example, ICMP messages are often suppressed by firewalls because ICMP messages can be exploited as a tool for scanning vulnerable ports and addresses. On the other hand, the later can be more reliable in that the detection mechanism focuses only on the scan behavior of attackers. However, the detector has to scrutinize the entire memory block to check the receipt of connection grant in determining the presence of attacks.

To solve this problem, we adopt the second approach in detecting failed connections but avoid exhaustive memory search by employing pessimistic failure counting and lazy detection techniques. The proposed scheme based on the connection failure may not detect attacks using UDP protocol. However, we believe that the proposed scheme can be extended to support UDP traffic with similar techniques used in [9] [10]. Therefore, we limited our focus on TCP flows in this paper.

In the pessimistic counting technique, we initially count all the connection attempts as failures and restore the connections as successes when the responses are received. Since the connection failure detection mechanism is built upon the destination...
address dispersion estimation technique, we explain the failed flow dispersion estimation with Figure 2 again. When a SYN packet arrives, the <destination address, destination port> is hashed to find the corresponding hash line. If the hash line is empty, it is updated with the current VTS ($vts_1$) and the failure counter $q_1$ is incremented by 1 (Case 1 in Figure 2). If the hash line has a different VTS ($vts_2$), then the previous VTS ($vts_3$) is backed up into the corresponding hash line of the other hash table in the dual-hash architecture (Case 3 in Figure 2). Without the backup mechanism, the latest connection failure information can be lost. For example, assume that there are consecutive requests for the same flow on $vts_1$ and $vts_2$ respectively. If the hash line is updated with $vts_2$, the information of unresponded connection request on $vts_1$ can be lost if the connection is granted within $vts_2$. Note that we only need to backup the connection failure occurred in the latest time bin due to the disjoint set condition. If a SYN/ACK is received, the VTS in the hash line is cleared to set the connection as a success, and then MWSM looks up the other hash table to restore the previously detected connection failure. All other packets and cases are ignored.

Although the pessimistic failure counting approach effectively avoids the exhaustive memory search and shows accurate estimation, a detailed evaluation exhibits over-estimation at the latest time bin. Due to the pessimistic counting, all the connection requested several milliseconds before the end of each time bin are counted as failures. The problem can be solved if the estimation is delayed for the round-trip time (RTT). Therefore, we use the estimated RTT in the lazy detection scheme, and we followed a typical estimation framework in the RTT estimation. To minimize the overhead come from RTT estimation, we randomly sampled flows (roughly 15%) and measured their round trip time. To allow some variations, we used $RTT_e = RTT_0 + 2 \times STD$, where $RTT_e$, $RTT_0$, and $STD$ stand for the estimated RTT, the average RTT of the sampled flows, and the standard deviation of RTT in the sampled flows, respectively. Although sophisticated estimation techniques may help in obtaining more accurate estimation, we did not delve into this issue because the sophisticated estimation may involve more overhead and the performance gain come from the improved estimation accuracy can be marginal in our study.

IV. PERFORMANCE EVALUATION

The proposed failed flow dispersion estimation scheme was evaluated through extensive simulations using real-world traffic trace files. We first investigated the estimation accuracy and memory consumption of MWSM. We then extended MWSM into a complete scan detector to study the impact of MWSM in the performance of attack detection. In all experiments, we set the interval of time bin ($\omega$) to 10 seconds and a maximum monitoring window size to 500 seconds as in MRDS [11]. In the simulations, we used three one-hour-long full payload traffic data gathered from a B-class network with an average traffic rate of 676Mbps. These traces were captured in May, June, and July 2008 and labeled by Trace M, Trace J, and Trace L respectively. Table I summarizes the captured trace data.

**Estimation Accuracy:** To study the accuracy of the proposed scheme, we examined the standard error (standard deviation of the ratio of the estimated flow dispersion and the bias (expected relative error) defined by $E(n/n) - 1$ as in [15], where $n$ is the estimated flow dispersion and $n$ is the measured flow dispersion. Figure 3(a) shows the standard error of failed flow dispersion with various hash sizes and monitoring windows under Trace J. The standard error initially shows large values with small monitoring windows, but it is decreased to less than 0.1 as the window size increases. Through traffic analysis, we found out that the main reason of the error comes from the flows with long RTT and such flows contribute to over-estimation of MWSM on short monitoring windows. Without such flows, the standard error becomes stable and successfully limited under 0.1. In addition, the standard error is still 0.18 (0.16 with the 512 hash size), less than 0.2, even in the worst case.

We believe that the impact of such error on the performance of port scan detection can be negligible since 1) fast scan attacks generally show order of magnitude larger scan rate compared to the normal flows, and 2) slow scan attacks can only be detected after long period of time. (after the estimation of MWSM is stabilized.) As the monitoring window size increases, the standard error slightly increases due to the hash collision, but the impact of the hash collision limited under 0.1 when the hash size is 512. Therefore, we set the hash size to 512 for the following simulations, and MWSM can accurately estimate up to 2500 flows within the standard error of 0.1 with this configuration [15]. The expected relative error also shows the similar trends with the standard error in Figure 3 (b).

**Memory Consumption:** We next investigated the memory consumption of the proposed MWSM technique. Unlike MRDS where the total memory consumption is determined by the number of internal hosts and the number of unique destination addresses contacted by each internal host, the total memory consumption of MWSM is determined by the number of internal hosts and the hash table size. We compared the memory consumption of MWSM with the 512 hash size and MRDS, since we configured the hash size of MWSM to 512 in the previous experiment. Table II summarizes the maximum memory consumption for each technique. For both techniques, the total memory consumption shows the largest value with Trace L since it has the largest number of internal hosts. However, MWSM shows much smaller memory consumption for all traces. On average, MWSM only consumes 0.94M bytes.
of memory whereas MRDS consumes 1.57M bytes.

**MWSM-based Scan Detection Performance:** To study the impact of the proposed technique on attack detection capability, we examined the trace files and identified the five attacks in our datasets, and the attacks were used as a ground truth to be compared against the alarms raised by the scan detectors. To identify attackers, we used Snort [4], Bro [5], MRDS [11], and the proposed technique, and manually examined the alarms raised by each system. The behavior of the identified attacks is described in Table III.

To extend MWSM into a complete scan detector, we followed the detection architecture and the configuration guideline given in MRDS [11], since both MRDS and the proposed scheme share the multi-resolution concept. Therefore, we configured MWSM with 13 monitoring windows, target scan rate from 0.1 to 5.0 and $\beta = 65536$ (in the conservative cost model). Here, we slightly modified the target scan rate to the half of the values used in MRDS because it is configured for destination address dispersion, not failed flow dispersion. In each simulation, we examined each trace data under both detection schemes, which was trained by the other two trace data, one at a time. Table IV shows the number of hosts detected by MRDS and the proposed MWSM scheme, respectively. Both detection techniques successfully detected all five attack instances in the simulations. However, with the detector extended from MWSM, the total number of false alarms is reduced from 216 to 87.

**V. CONCLUDING REMARKS**

Although the multi-resolution approach (MRDS) is believed to be effective in detecting a wide range of scanning attacks, a detailed performance study showed limitations such as high false positives in the presence of the greedy applications, unpredictable memory consumption, and exhaustive memory search. In this paper, we proposed a novel failed flow dispersion estimation technique, called Multi-Window State Map (MWSM) to overcome the problems. The proposed MWSM scheme uses dual hash architecture with the virtual time stamp (VTS) to make the failed flow estimation and multiple window monitoring simple. The hashing helps in reducing per packet operation whereas VTS helps in simplifying the multi-window monitoring. Thanks to its simple architecture, we believe that the proposed technique is well suited to hardware implementation and thus is practically viable in modern high speed intrusion detection systems.

Extensive performance evaluation with real-world traffic traces indicated that the proposed MWSM scheme accurately estimates the number of failed connections. In addition, the scan detector with MWSM reduces the number of false positives by 60% compared to MRDS. Our future work will involve the fine tuning of a MWSM based port scan detector and hardware implementation of MWSM to study the effectiveness of MWSM in the high speed networks.

**REFERENCES**


