Scan Detection - Revisited

Levent Ertöz[1], Eric Eilertson[1], Paul Dokas[1], Vipin Kumar[1], and Kerry Long[2]

[1] University of Minnesota, Minneapolis MN 55455, USA
[2] Army Research Laboratory, Adelphi MD 20783, USA

Abstract. Although scan detection can be a very important tool for a security analyst, its value is somewhat overlooked in the security community. One reason is that good tools for doing proper scan detection to this point simply do not exist. The existing schemes essentially look for IPs that make more than X connections in Y seconds. If the threshold is kept too high, than such schemes find high volume scans, but are unable to find stealthy, or low volume scans. If the threshold is made too low, the schemes suffer from a high false alarm rate. This paper presents new scan detection techniques that have much lower false alarm rate and much higher coverage than existing techniques. Some of these techniques are particularly suitable for detecting very slow stealthy scans. The paper also presents a detailed experimental comparison of our new and existing scan detection methods on a large size network data from the University of Minnesota. These experimental results show that the proposed methods have much better recall than the basic time window based schemes, and are able to achieve a near-zero (less than 1%) false positive rate. A scan detection system incorporating these new techniques has been in production use over the past year at the University of Minnesota as a part of the Minnesota Intrusion Detection System (MINDS) and at the US Army Research Laboratory Center for Intrusion Monitoring and Protection (CIMP) that analyzes traffic from many DoD sites around the country. In particular, the ability of our new schemes to detect very low volume scans has lead to the identification of several compromised computers used as stepping stones inside the Army's network, compromises that were not detected by any other method.

1 Introduction

A precursor to many attacks on networks is often a reconnaissance operation, more commonly referred to as a scan. Identifying what attackers are scanning for can alert a system administrator or security analyst to what services or type of computers are being targeted. Knowing what services are being targeted before an attack allows an administrator to take preventative measures to protect the resources they oversee, e.g. installing patches, firewalling services from the outside, or removing services on machines which do not need to be running them.

Intrusion analysts, whether monitoring networks for the government or the private sector, are required to perform far more analysis tasks then time will allow. Because of this, only a small subset of the alarms that various intrusion

tools produce can be investigated further. Analysts quickly discover that it is most effective to investigate intrusion alerts that indicate immediate and catastrophic compromise. This leaves very little time to analyze alerts such as scans that are essentially informative in nature but do not necessarily indicate immediate compromises. Hence, despite the importance of scan detection, its value is somewhat overlooked in the security community. One reason is that there is a lack of good tools for doing proper scan detection.

The existing scan detection schemes essentially look for IPs that make more than X connections in Y seconds. These schemes are very good at picking out disperse noisy scans. Unfortunately, tools based on these techniques are quite bad at detecting stealthy scans or scans targeted specifically at the monitored enterprise - the type of scans that analysts would really be interested in. Stealthy scans can be defined as scans that would normally not trigger typical scan alert technology. The adversary is keenly aware that most scan detectors are set to alert based on the number of connections attempted by a given host over a predefined period of time. It is fairly trivial for an adversary to adjust his scans to evade detection by slowing down the frequency of his transmissions. Intrusion analysts and several IDS vendors tried to counter this threat by either reducing the connection threshold and/or increasing the time window threshold in their scan detection technologies. This created an untenable situation where chatty yet benign protocols (e.g. protocols such as netbios that talk to many hosts in a relatively short time) generating numerous alerts and effectively disguising the presence of the stealthy scans. The result is the current situation where analysts either do not run scan detectors or essentially ignore their outputs in the interest of time.

This paper presents new scan detection techniques that have much lower false alarm rate and much higher coverage than existing techniques. A key strength of these new scan detection techniques is their ability to detect stealthy scans. This paper presents some heuristics to reduce false alarm rates of the existing schemes, as well as a new way of detecting scans which makes use of usage information. The paper also presents a detailed experimental comparison of existing and our new scan detection methods on a large size network data from the University of Minnesota. These experimental results show that the proposed methods have much better recall than the basic time window based schemes with a near-zero false positive rate. These new techniques have been in production use at the University of Minnesota and at the US Army Research Laboratory Center for Intrusion Monitoring and Protection (CIMP) that analyzes traffic from many DoD sites around the country. The ability of our new schemes to detect very low volume scans has lead to the identification of several compromised computers used as stepping stones inside the Army's network, compromises that were not detected by any other method.

2 Related Research

Scan detection has been often thought as the process of counting X number of events in Y number of seconds. The most widely used scan detection method counts the number of unique destination IPs talked to by each source on a given port in a given time window. If this count within a time window of Y seconds is greater than a user specified threshold X, then the source IP is considered to be scanning. We will refer to this as the basic scheme. This method requires storing information about the destination IP and port as well as the time stamp. This method has a reasonable false alarm rate. This basic method is used in SNORT's [1] scan detection module, which detects host scans and port scans. However, this method is not suited for detecting slow scans. To do so would require having a large time window while keeping the threshold low (which also increases memory requirements considerably) to increase the recall. But this will also result in a higher false alarm rate. Typical false alarms generated using this scheme are web browsing, crawlers, sending e-mails to large mailing lists, peer-to-peer applications, etc.

One standard way to handle slow scans is to add a connection window which keeps a history of the most recent N connections made by each source to a destination port. Using only a time window to detect slow scans, requires storing significant amount of data, most of which are related to legitimate sources that make many connections. Using a connection window allows us to store enough information about low volume talkers to detect slow scans while minimizing the storage for high volume talkers.

The basic method and its derivatives do not take into account the usage information for the hosts that are talked to. An external source connecting to a server contributes just as much to the scan score as another external source trying to connect to an internal machine where the service doesn't exist. Obviously, the latter one should be weighted more than the former. The method used in SPADE [2] which is available as SNORT preprocessor, makes use of the usage statistics. It maintains the joint probability distribution of destination port and destination IP, which they have found to work the best among different methods. It raises an alarm on a packet if the negative log likelihood of the destination IP / port combination for a packet exceeds the threshold [1]. Using the methodology describe above, SPADE raises too many alarms and it is turned off by default in SNORT. For example, SPADE will raise an alarm on the first few connections to previously unused IP / port combination, even if this is the only activity from that source. For example, connections from a single source to a rarely used service on a single IP will be declared as a scan simply because it is rare. This problem is exaggerated due to the fact that the history cannot be kept for long periods of time. As a result, every occurrence of low intensity periodic events can potentially be declared as scans. In addition to the false alarms, SPADE may also miss very popular scans (i.e., the scans that are performed by many

sources) simply because the corresponding counts in the usage statistics will be high.

Another common approach tries to address the false positive problem in scan detection by only looking at the ICMP replies to packets sent to unallocated/dark IP spaces. No legitimate connection should involve a dark IP address. Other than misconfigured machines and rare occurrences of users typing IP addresses wrong, the rest of the connections to dark IPs should be due to scans. This approach is good for detecting random scans, as they have a high probability of hitting unallocated IP ranges. However, this methods cannot detect smarter scans (e.g. those that avoid unallocated IP ranges by using BGP information). Other problems with this method include not knowing which allocated IPs were scanned on the inside network and the inability to identify computers that responded to scans. In addition, scanners that randomly pick IPs to touch will become less effective and thus less common with the move from IPv4 to IPv6; hence we need new techniques to be able to detect smarter scans.

3 Proposed Method

In this section, we present some techniques that address the limitations of the basic method.

3.1 Incorporating Heuristics in the Basic Method

To address the high false alarm rates of the basic threshold based schemes, we make use of two characteristics that differentiate scanning behavior from normal traffic. First, most scan traffic is connections to ports on which no service is running. These connections will not contain more than 3 packets, as the three-way-handshake cannot be established. Even for ports on which a service is running, the scanner will try to terminate connections quickly to maximize the number of machines they can scan in a given time. In contrast, majority of legitimate traffic tends to last longer than 3 packets (approximately 60% of all connections involed more than 3 packets). Hence, the first heuristic we use is to only consider incoming flows that are less than 4 packets. This heuristic helps eliminate a large fraction of normal traffic in scan detection, which potentially decreases false alarms.

Second, normal traffic can consist of short connections from one IP to many other IPs (raising false alarms on basic threshold based schemes discussed earlier). For example, during typical web browsing, a user might do a web search and visit many sites returned in the search results in a short period of time.

[1] Maintaining statistics on the IP / port combinations require considerable amount of memory. For example, for a class B network, the number of IP / port combinations inside the network is $2^{16} * 2^{16} = 2^{32}$. Storing 4 bytes per entry would require 16 GB of memory for a relatively small sized network. To address this problem, in SPADE, usage statistics are stored in a sparse matrix representation and the counts are aged and an entry is deleted when the count falls below a certain threshold.

However, most of these connections tend to be on random IPs not necessarily concentrated on a specific subnet. In contrast, most scan traffic tends to make connections to many machines within same subnets. This is particularly true for targeted scans that look for vulnerabilities in a specific organization's network. Hence, the second heuristic we use restricts scans into subnets. This is done by counting the number of connections by a source to IP/port combinations on a network. This reduces false alarms by not alerting on typical normal traffic.

3.2 Usage Information

Here we present a scheme that incorporates complementary strengths of the basic scheme and the scheme used in SPADE, while avoiding their weaknesses. As discussed earlier, SPADE can raise an alarm on a packet / connection regardless of what else is done by the source IP. But it does give higher importance to destination IP / port combinations that are rarely used. On the other hand, the basic scheme considers all the traffic generated by a source IP in a given time window before declaring it a scanner. However, the basic scheme does not differentiate between connections to heavily used servers and previously unseen traffic. Our proposed scheme not only looks at recent connection history for each source, but also weighs each of the touches to unique destination IP / port combinations differently in its scoring according to the usage information.

In our proposed scheme, just like in SPADE, we maintain statistics for destination IP – destination port combinations. We would also like to detect outbound scans as they may point to infected hosts in our network. Therefore, we have to maintain statistics for outside hosts as well. We use these statistics to improve on the basic method. Instead of incrementing the scan count for a host by one every time it touches a new IP on a given port, we increment the scan score by

$$\frac{1}{1 + lg\left(count_{IP/port}\right)} \tag{1}$$

By doing so, we can reduce the false positives that might be generated by the basic method. For example, local users browsing various websites to get the daily news will be weighed lower since the volume of the traffic on that IP / port combination will be much higher compared to a random IP/port combination. It would take many more touches on commonly used combinations as opposed to few touches on infrequent combinations for a host to be declared a scanner.

As we stated earlier, the basic method cannot detect low volume and stealthy scans. We partly address this by making use of usage statistics. Moreover, we also extend the detection capabilities of the basic time window method by using the concept of a connection window, that is, we not only score connections made by a host in a given time window, we also score last N connections made by a host. Using the connection window, we keep history information for low volume hosts longer. This allows us to detect low volume scans as well. A host is declared to be a scanner if the sum of the scores according to equation 1 for the unique IP addresses touched on a given port in a time / connection window exceeds the scan threshold.

Representing and storing the usage information exactly for each IP / port combination requires too much storage and is not feasible. Specifically, even the complete representation of port / IP combinations within a class B network will require 16 GB of memory. Complete representation of port / IP combination statistics for the entire Internet will require a staggering $2^{32} * 2^{16} * 4 = 2^{50}$ bytes. To bring down the memory requirements to store these statistics, we do not keep statistics about each individual host and port, but we group hosts and ports into blocks as can be seen in figure 1. When we group IP addresses into blocks, we would like to preserve the geographic locality by keeping high order bits. We would also like to preserve some low order bits in order to be able to detect scans on subnets. Therefore, we chose to group IP addresses by keeping *num_high_bits* plus *num_low_bits* bits. We also grouped the unprivileged ports into blocks of *port_resolution* and treated privileged ports individually. Depending on the memory available, these parameters can easily be adjusted to bring down the memory required for maintaining the statistics.

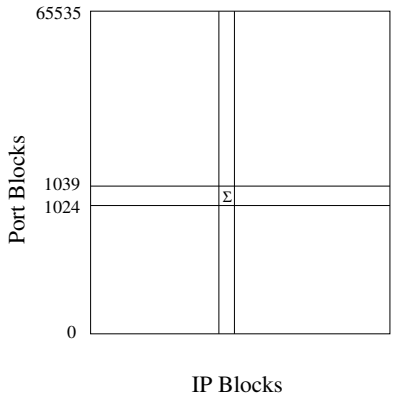


Fig. 1. Aggregated Usage Statistics

In the next section, we will present experimental comparisons between the basic scheme, the basic scheme improved using heuristics and the proposed scheme.

4 Experimental Evaluation

In our experiments, we used netflow data collected at the border router at the University of Minnesota. The evaluation was done on a half hour worth of data from midnight to 0:30 AM. The main reason why the evaluation wasn't done for a time window during day time is the amount of manual effort needed to label the scan detection output. The number of flows recorded during day time can be as many as three times larger than the number of flows collected during night time. Since we are only evaluating inbound scans, the data from the users

at our University would not have affected the results much. The data collected consisted of 3,943,410 netflows, the data transfer rate at this link was 221 Mbps. The breakdown of data transfered per protocol is given in table 1.

Protocol	Flows	Octets	Packets
TCP	2317461	48030621617	71794984
UDP	1525076	1557138198	10280879
ICMP	98885	25944042	244081
IPv6	1175	2920908	7714
ESP	567	257380176	271262
PIM	121	319666	3596
IGMP	47	4172	149
OSPF	26	208980	2540
AH	24	39032	189
GRE	22	299411	1053
IP	6	920	23

Table 1. Protocol Distribution

We had to do a pre-processing step due to the nature of the netflow data. Flows are unidirectional; a TCP session will result in two unidirectional flows. We paired up the TCP flows into sessions using the 5-tuple (IPs, ports and protocol) and the time stamp information. After determining the client-server relationship for the TCP sessions, only clients are considered in scan detection. Not considering servers in scan detection eliminates the possibility of declaring the servers to be scanning. This implementation has a hard timeout on the connection window; connections from sources are removed from the history after the hard timeout even though it may not have reached N connections. Also, scans detected using the time window are not considered for evaluation in the connection window.

We will use B_k to denote the basic scheme, H_k to denote a scheme that incorporates both of the previously described heuristics, and U to denote the usage based scheme, where k is the threshold. We applied different methods of scan detection with different thresholds. The first two methods correspond to the basic method which simply looks for X events in Y seconds. We used a 10 second time window, as well as the connection window extension with a width of 256 connections. Both of the thresholds were set to 10 in the first experiment and 5 in the second. The next two experiments used the previously described heuristics, only considering scans to a network and the number of packets in a flow less than 4, using the same thresholds in the first two experiments. In the last experiment we again used the heuristics as well as the usage table as previously described. We combined 8 high bits and 4 low bits of the IP to define the IP blocks, and used single ports below port 1024 and groups of 16 for high ports for which the counts are collected. For this experiment, we used a threshold of 1.01. This meant that the scan had to touched at least 2 unique IPs as the

contribution from a first touch to a never before used block in the usage matrix is 1.

In these experiments, we only investigated inbound scans due to privacy considerations. Reported outbound scans were filtered out before our analysis of the output. In the production mode at the University of Minnesota and at ARL CIMP, it is used to catch outbound scans as well, many of which identify compromised machines not easily identifiable by other means.

After running the experiments, the security analyst on our team labeled the output. Many of these were obvious scans where a source touched many ports on many IPs. Many of the scans were easily labeled because they attempted to touch multiple IPs where either port was blocked at the border or the IP was not allocated. After labeling the obvious scans, we were left with approximately 20% of the scans which required investigation. This investigation involved a number of steps, including checking of the IP address to see if it resolves to a name, checking if the service is running on the destination that is scanned, and investigating other IPs touched by the suspect source IP over a larger time window. Traffic from several web crawlers (e.g. googlebot, inkotmi) were detected as scans in some experiments. They were labeled as false alarms since they are not trying to connect to machines in order to verify the existance of a service. It is interesting to note that almost half of the scans found were due to Slammer worm still in existance.

The definition of a single scan we adopted for this evaluation is a source IP attempting to connect to the same service. Using the heuristics, scans were reported based on a network block (obtained from the router, available in netflows). If multiple scans from the same source to the same service were reported for multiple network blocks, these were aggregated into a single scan. In the following tables, the number of scans found using time and connection windows may add up to more than the total number of scans reported. This is due to the fact that scans by one source for a particular service to one network block may have been detected using the time window and another using the connection window.

Table 2 shows the number of scans detected correctly and number of false alarms by each method broken down by time and connection window. Detections using time window has a much lower false alarm rate but has significantly less coverage. Using connection window between 5 to 7 times as many correct scans were reported. However, the false alarm rate increased substantially.

If we used the basic scheme without the connection window extension, we would have only detected 250 scans correctly with 5 false alarms. If we lowered the threshold from 10 to 5 the coverage increases to 315, but the number of false alarms increase to 70. Using time and connection window with a threshold of 10, we detect a total of 1626 correct scans with 198 false alarms, and with a threshold of 5, we detect a total of 2459 scans correctly with 3638 false alarms. Applying the heuristics to the basic method, reduced the coverage slightly while greatly reducing the number of false alarms. With the usage based method, the coverage

increased primarily due to the detection of stealthy scans while maintaining a low false alarm rate.

Most of the false alarms in the basic scheme (with a threshold of 5) are due to Blubster (1470), Gnutella (1122), Web browsing / crawlers (561) and E-mail (202) connections. Increasing the threshold to 10 reduces the number of false alarms to only 142 for Web and 18 for E-mail and completely eliminates the Blubster and Gnutella alarms for this data set. The drawback of raising the threshold is a significant drop in coverage. By using the heuristics, most of the false alarms can be avoided as can be seen in table 2. Several of the false alarms for each detection method were due to replies to Network Time Protocol (NTP) requests from distinct sources. These false alarms can easily be avoided by matching these requests and responses. The number of false alarms which this would have removed are Usage:6, H_5:3, H_{10}:3, B_5:16 and B_{10}:9.

We also compared the false alarms generated by H_5 and Usage. The false alarms generated by the usage method do have low counts (less than 5) and they do not overlap with the false alarms generated by H_5. Even though the sources of scans reported by H_5 touched at least 6 unique IPs, the usage method was able to eliminate the false alarms since the connections were to commonly used IP / port combinations.

Method	TP_{time}	FP_{time}	TP_{conn}	FP_{conn}	Total Correct	Total False[1]
U	292	2	2294	20	2561	22
H_5	314	1	2105	6	2419	6
H_{10}	256	0	1329	4	1585	4
B_5	315	70	2144	3568	2459	3638
B_{10}	250	5	1376	193	1626	198

Table 2. Detections and false alarms

In table 3, we compare the coverage of scans between different experiments. Entry in row i and column j represents the number of correct scans found in experiment i and not found in experiment j. For example, there are 252 scans correctly detected by the Usage method and not detected by H_5. The greatest coverage is obtained by making use of the usage statistics. However, this method and the methods using the heuristics miss some of the correct scans detected by the basic scheme with a threshold of 5. This is due to sources hopping from one network to another and only touching very few machines in each block within the University. In addition, the usage method can miss real scans if the corresponding entries in the usage statistics table have high counts for the connections involved in the scan. For example, a web server will not only cause connections to that server to have very low contributions but also will make connections that map into the same entry in the usage table have low contributions. One

[1] Some of the false alarms are due to not matching UDP requests with replies. These can trivially be removed by building "UDP sessions".

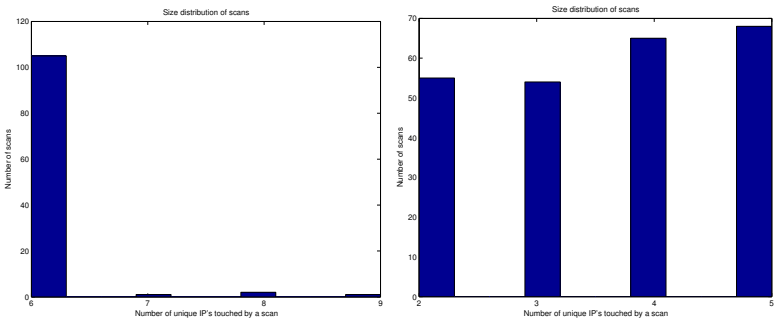
way to address this problem is to maintain multiple usage tables with different grouping functions, i.e. using every other bit or using 8 high bits and 8 low bits or some other combination. Legitimate connections will usually fall into blocks with high counts because they tend to connect to highly used servers. Scan connections may avoid detection in one mapping because of the blocking (e.g. by web servers), but will most likely be detected using one of the many alternate mappings.

Method	U	H_{10}	H_5	B_{10}	B_5
U	0	976	252	988	267
H_5	109	833	0	848	20
H_{10}	0	0	0	14	13
B_5	164	886	60	856	0
B_{10}	53	55	56	0	24

Table 3. Coverage comparison of different methods

Some scans detected by the H5 method are not detected by the usage method and vica versa. Figure 2 shows the size distribution of scans detected by only one of the methods. All of the scans detected by the usage method and not by the H5 method are slow scans; they involved less than or equal to 5 IPs. We did a separate experiment to investigate what it would cost for the H5 method to catch those scans as well. We reduced the scan threshold from 5 to 4. The number of correctly detected scans went up from 2419 to 2801. However, the number of false alarms increased dramatically from 6 to 660. One reason for this increase is that, commonly used servers at the University do not have a single IP, but there can be 5 IPs mapping to the same DNS name. Even though a user is using a service on a particular URL, the user might actually be touching 5 IPs. This shows that without making use of the usage information, we cannot detect slow scans with a low false alarm rate.

Even though the coverage of the usage method is better than any other methods we tried, there are still scans detected by the other methods and not by the usage method. For example, there are 164 scans correctly detected by the basic method that are missed by the usage method. Since usage statistics are not stored in exact detail, there will be some cases where a scanner is touching IP / port combinations that map to frequently used blocks in our table. In such cases, those scans will effectively be masked. To avoid this problem, we performed another experiment where we used two different mappings for the IPs and maintained two separate tables. If the score for an IP / port combination rises above the thresholds using either of the usage tables, then it is declared to be a scan. By doing so, the number of correctly detected scans increased from 2561 to 3082 and the number of missed scans that are detected by the basic method decreased from 164 to 60. In the original experiment, the usage method generated 22 false alarms, 6 of which were due to NTP. By using two different mappings at the same time, the number of false alarms increased from 22 to



a.–Size Distribution of Scans found by H5 and not by Usage

b.–Size Distribution of Scans found by Usage and not by H5

Fig. 2. Coverage comparison of Usage and H5

35. Six of these new false alarms are due to ICMP replies which can safely be ignored (host unreachable messages cannot be a part of a scan).

Table 5 shows the commonly scanned ports, how many times they have been scanned for, the average size of the scan and their detection method. These statistics are calculated using the results of the usage method as it has the broadest coverage.

5 Conclusion

In this paper, we presented new techniques for scan detection that address the shortcomings of the traditional scan detection methods. These schemes, implemented in MINDS, effectively increase the time window threshold over which scans are detected, reduce the number of connections threshold which triggers an alert, while at the same time eliminating many of the false positives that benign protocols yield. Analysts can be reasonably sure that if MINDS alerts on a scan it is indeed a scan. Analysts no longer need to ignore the indication of a slow scan. For example, an indication by MINDS that one of Microsoft's netbios ports is involved in a stealthy scan can be taken more seriously. Previously an alert for a Microsoft scan was often dismissed, as normal network traffic on these ports frequently triggered the scan detection tool. Now analysts using MINDS at the ARL CIMP and University of Minnesota find that though some indications of scanning activity on Microsoft's netbios ports are still benign, the majority of the time they are not. The benign activity only involves a small set of computers while the set of targets scanned extend beyond that set. Since the analyst is not likely to take the time to consistently investigate the output of scanning tools, having an implicit trust in the output of the scanning tool is of critical importance.

Source IP	Port / Protocol	Number of unique IPs touched	Score	Detection method	Start time	End time
63.231.x.245/12	389/udp	2	1.18	time window	0306.00:07:32.777	0306.00:07:32.877
63.239.x.6/24	9370/udp	4	1.01	connection window	0305.23:59:41.382	0306.00:09:55.625
66.41.x.22/20	389/udp	2	1.18	time window	0306.00:18:57.268	0306.00:18:58.268
67.154.x.2/14	524/tcp	3	1.68	connection window	0306.00:06:31.320	0306.00:25:43.315
67.154.x.120/14	524/tcp	4	2.60	connection window	0306.00:06:52.288	0306.00:25:01.178
128.111.x.12/16	123/udp	6	1.58	connection window	0306.00:00:25.382	0306.00:25:44.767
132.163.x.101/16	123/udp	5	1.56	connection window	0306.00:02:17.479	0306.00:26:40.871
132.163.x.103/16	123/udp	5	1.62	connection window	0306.00:02:17.479	0306.00:26:40.871
155.68.x.128/16	6346/tcp	2	1.18	connection window	0306.00:18:24.244	0306.00:23:14.466
158.152.x.222/16	6346/tcp	3	1.36	connection window	0306.00:12:44.682	0306.00:24:21.886
163.120.x.97/16	6346/tcp	3	1.02	connection window	0306.00:02:05.643	0306.00:22:39.261
166.90.x.130/16	2048/icmp	3	1.07	connection window	0305.23:59:05.061	0306.00:23:21.102
169.237.x.1/16	2048/icmp	2	1.18	connection window	0306.00:13:35.355	0306.00:16:30.627
170.215.x.18/21	524/tcp	4	4.43	connection window	0306.00:02:39.163	0306.00:28:16.883
192.5.x.41/24	123/udp	43	11.80	connection window	0305.23:33:18.719	0306.00:27:49.799
192.5.x.41/24	123/udp	11	3.44	connection window	0305.23:57:51.262	0306.00:23:44.762
192.5.x.41/24	123/udp	9	2.39	connection window	0305.23:58:35.609	0306.00:27:51.295
192.5.x.41/24	123/udp	10	3.14	connection window	0305.23:47:34.156	0306.00:23:36.243
192.5.x.209/24	123/udp	40	11.46	connection window	0305.23:40:03.485	0306.00:27:49.799
192.5.x.209/24	123/udp	12	3.06	connection window	0305.23:50:09.081	0306.00:27:38.831
192.5.x.209/24	123/udp	11	3.36	connection window	0305.23:33:18.315	0306.00:26:32.219
192.5.x.209/24	123/udp	9	2.29	connection window	0306.00:04:37.903	0306.00:27:33.975
192.102.x.251/24	0/icmp	3	1.26	connection window	0306.00:08:06.281	0306.00:16:51.992
199.109.x.13/24	781/icmp	2	1.18	connection window	0306.00:22:14.289	0306.00:25:08.290
203.197.x.129/24	2048/icmp	3	1.15	connection window	0306.00:06:23.288	0306.00:22:56.429
204.176.x.5/24	2048/icmp	3	1.07	connection window	0306.00:23:09.285	0306.00:26:39.163
207.46.x.100/18	123/udp	12	2.04	connection window	0306.00:00:00.358	0306.00:18:36.404
207.46.x.100/18	123/udp	11	1.87	connection window	0306.00:01:28.183	0306.00:26:45.619
207.46.x.100/18	123/udp	25	4.26	connection window	0306.00:08:00.773	0306.00:16:19.615
221.233.x.244/14	113/tcp	2	1.18	connection window	0306.00:25:10.402	0306.00:25:25.563

Table 4. False Alarms - Usage Method

References

[1] www.snort.org

[2] Stuart Staniford and James A. Hoagland and Joseph M. McAlerney: Practical automated detection of stealthy portscans J. Comput. Secur. **10** (2002) 105–136

Port	Number of Distinct Scanners	Average Size of Scan	Distinct $Scanners_{time}$	Avg. Size of $Scan_{time}$	Distinct $Scanners_{conn}$	Avg. Size of $Scan_{conn}$
1434	1061	44.0386	23	1290.65	1038	16.42
445	549	32.0055	2	2433	547	23.23
3127	326	89.6472	9	103.33	318	88.98
135	246	148.76	9	3381.67	237	26
137	213	143.174	197	154.51	16	3.56
1080	37	118.946	2	1013	37	64.18
3128	35	62.2286	0	0	35	62.22
139	16	1499.56	2	10925	14	153.07
2048	12	1810.75	10	2155.9	4	42.5
80	8	49.125	1	32	8	45.1
9898	3	193.667	3	193.67	0	0
443	3	6997	2	10431.5	2	64
1433	3	4294.33	2	6401	1	81
8080	2	1064	2	1015.5	2	48.5
79	2	1078	2	1016	2	62
6000	2	1067.5	2	1019	2	48.5
53	2	1065.5	1	2006	2	62.5
515	2	1064.5	1	2047	1	82
514	2	42	1	8	1	78
3389	2	1066.5	1	2008	2	62.5
3306	2	1065	1	2004	2	63
23	2	1076.5	1	2006	2	73.5
22	2	1066.5	2	1033	2	33.5
161	2	1068	1	2054	1	82
143	2	1062.5	1	2004	2	60.5
111	2	1080	2	1080	0	0
110	2	59	2	20	1	78

Table 5. Commonly scanned ports