Proceedings of the 2005 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY June 2005

Host Anomalies from Network Data

Carrie Gates [†], *Member, IEEE*, and Cpt. Damon Becknel [‡]

Abstract – Network administrators need to be able to quickly synthesize a large amount of raw data into comprehensive information and knowledge about a network system in order to determine if there is any unusual activity occurring on that network. This paper presents some initial results of a simplistic baselining method applied to a class B-sized network. These baselines are then used as the basis for an anomaly detection system that examines unusual amounts of activity to any one port on any one host. Thus we provide a system that can detect changes in the activity of any one host, regardless of whether those changes are noticeable when observing overall traffic behavior.

Index terms – network intrusion detection

I. Introduction

Intrusion detection is an important part of an information assurance strategy, since identifying machines that have been compromised in turn identifies documents, communication channels and workflows that may have been compromised. While there are host-based intrusion detection systems, the majority of deployed systems operate at the network level (e.g. Snort [1], Bro [2], MINDS [3]). The approaches used by these systems each have their short-comings. In the case of Snort, which uses a signature-based intrusion detection approach, a signature for the exploit used by an adversary must already be written and added to the rule-based. In the case of MINDS, where a more anomaly-based approach is taken, only network-level anomalies are detected, without specifically identifying changes in behavior at the host level.

This paper addresses the gap between intrusion detection approaches. We describe an anomaly-based system built using simple statistical tests that detects changes in host-level behavior, while still operating at the network level. Additionally, rather than requiring packet-level analysis, our system is based on flow-level analysis, allowing for high-bandwidth, high-utilization networks to more easily be monitored. We base our system on the assumption that malicious activity can be identified based on changes in port usage at the host level (as opposed to the network level), and that this can be used to identify possible compromised machines.

In Section 2 we discuss related work in the intrusion detection field. In Section 3 we describe our system in detail, and provide experimental results from deploying this approach on a /16 subnet in Section 4. We discuss the limitations of this approach in Section 5, followed by a concluding section.

II. Background

Denning [4] was the first to propose anomaly-based intrusion detection, based on the assumption that malicious behavior will be anomalous in any given system. The earliest example of this approach is Network Security Monitor (NSM) [5]. NSM generated a matrix of connection information, and then compared the patterns observed in the matrix to known signatures of malicious activity. In addition, they used a probabilistic analysis to determine normal, and alerted on any traffic outside of this pattern. They observed at the time that most hosts communicated with a very small number of hosts and services. However, it is likely that this pattern does not exist as strongly today, especially with the advent of the web and peer-to-peer applications.

Since NSM, several detection systems have been developed, with a focus on applying data mining techniques to classify network attacks [6]. In addition, anomaly-based methods, where the focus is to detect novel attacks, have also been developed. One example of this is the Minnesota Network Intrusion Detection Systems (MINDS) [3]. MINDS operates by clustering data and then for each point calculating how far it is from the cluster. If the point is an outlier, it is considered anomalous and an alert is generated. One of the challenges identified by Eilertson et al. for the MINDS system was profiling of individual IP addresses [7]. Their planned approach to this involved developing high-performance computing-based solutions.

[†] *Faculty of Computer Science, Dalhousie University, and CERT Network Situtational Awareness, Software Engineering Institute, Carnegie Mellon University*
[‡] *United States Military Academy, West Point*

ISBN 555555555/$10.00

Leckie and Kotagiri, in 2002, presented a novel approach to detection that was based on probabilistic modeling of port connection information [8]. However, they focused on using this approach solely for scan detection and not for more general intrusion detection. For each IP address in the monitored network a probability is generated that represents how likely it is that a source will contact that particular destination IP, $P(d|s)$ where d is the destination and s is the source, based on how commonly that destination IP is contacted by other sources, $P(d)$. A similar approach is used for each port. The probability $P(d)$ is based on the prior distribution of sources that have accessed that host, which implies that if the probabilities for this approach are generated based on a sample of network data, that if the monitored network is scanned regularly or heavily then the resulting distributions will include scans as being normal traffic. This will likely make the approach less accurate in practice.

Kim et al. also took a statistical analysis approach to intrusion detection, and also focused specifically on port scanning [9]. They generated a model of normal network traffic, and then searched for anomalous traffic that would indicate the presence of a port scan. This was loosely based on the footprint information that would be seen, as this pattern of activity would result in different statistical values. They performed four types of tests, divided into static and dynamic tests and employing z-tests and χ^2 tests. Their results indicated that these approaches were promising. However, they have not yet tested their approach on actual network traffic, but only through simulations. In addition, they assume that network traffic follows a Poisson distribution, when it is now widely believed that network traffic has a heavy-tailed distribution.

Iguchi and Goto perform a baseline method similar to ours, where they baseline the activity of different ports in order to detect anomalous activity [10]. Their baseline consists of creating a basic profile where a frequency distribution is stored for each port. Several distributions are generated, such as incoming packets per session, outgoing packets per session, incoming bytes per session and outgoing bytes per session. In addition, these profiles are aged over time, in order to accommodate natural network fluctuations. However, this approach has only been deployed at the border router, and has not been designed for observing each host in a large network, where scalability is more of a concern. As a result, they might not necessarily observe unusual traffic to a single host. In addition, they found that their approach worked well for well-known server ports, but was unable to be applied to ephemeral ports, whereas we are trying to monitor both in order to detect unusual events such as MyDoom [11] scanning.

III. Our Approach

We have developed an approach to anomaly detection that is based on simple statistical modeling at the host level. However, we perform this analysis at the network level and apply it to each host, thereby allowing a network administrator to determine if any individual host is experiencing unusual activity. Like other anomaly-detection systems, we assume that malicious usage will manifest itself in unusual traffic patterns.

While we are interested in identifying anomalies at the host level, we use the network level for two reasons. First, it cannot be assumed that users will monitor host-based logs and alerts for signs of intrusion. While having a network or security administrator responsible for monitoring each host avoids this issue, the administrator can then be overwhelmed by the number of hosts for which there are logs. Given this, we provide an analysis at the network level where the purpose is to identify any hosts that exhibit some change in characteristics. Second, a network-level viewpoint can provide more situational awareness. At the host level, an administrator only knows if that single host is experiencing unusual behavior. However, at the network level an administrator can observe if there are multiple hosts that have unusual activity, if the unusual activity amongst the hosts is the same or different, and if those hosts fall in some logical group (e.g. same subnet, same department).

Unlike most other systems, we use flow data rather than packet data. This allows us to more easily process information in high-bandwidth high-utilization environments as the network data has already been aggregated for us. In particular, we extend the SiLK [12] collection system and suite of tools (available at SourceForge [13]), which processes Cisco NetFlow version 5 [14]. Cisco NetFlow aggregates the traffic into a flow record based on address and temporal similarities, resulting in a record that contains routing information, source and destination IP and port information, and payload summary information (e.g. number of packets, number of bytes). We note that our approach should be easily extendable to packet-level analysis.

Our system consists of two phases: a training phase and an operational phase. The training phase is required because we assume no a priori knowledge about the hosts on a network. Rather, we observe network traffic and based on this information determine if a machine exists at a particular IP address and if it exhibits behavior that is characteristic of a server or a client. Once we have developed a baseline of typical activity, we enter the operational phase. This consists of comparing any given hour of data against the current baseline to determine if there is any unusual activity.

ISBN 555555555/$10.00

A. Training Phase

During the training phase, the system reads flow records and uses the data to calculate simple statistics. This step is meant to be performed on a set of (recent) historical data. The first step in the algorithm reads this set of data in one hour chunks and stores the number of flows observed for each port on each host. In order to reduce the memory footprint, the data structure that we use consists of an array of 65536 pointers to another array of 65536 pointers. These pointers represent every possible IP address. If a host is found to exist, then the appropriate pointer has space allocated to it to hold a structure that consists of a bitmap of 2048 32-bit integers and a pointer to a linked list. Each bit in the bitmap represents one of the possible 65536 ports, and is turned on to indicate that a particular port has seen at least one flow. The linked list consists of the number of flows for each port that had communication, where the order of the items in the list is the same as the order of the ports that have seen activity. For example, if the only ports who had any flows were 80 and 443, then the first item in the linked list would hold the number of flows for port 80, and the second for port 443.

Once all of the training data has been read in, the mean and standard deviation for each port on each host is calculated. We use the same data structure that was described above, however each node in the linked list now contains the average and standard deviation for each port that had flows on it. Additionally, the sum and sum of squares are kept, so that the structure can be easily updated with new information. The resulting data is saved to a file and used in the operational phase.

In addition to statistical information, the node for each host also contains a flag indicating if the host is a server or a workstation. All IP addresses receiving more than 100 flows/hour (on average) to a port less than 1024 are considered servers, and all other hosts are then considered to be workstations by default.

B. Operational Phase

This phase has been designed to work on one hour increments of data. One hour of NetFlow data is read into a structure containing a count of all flows observed by all ports on all hosts, the same as was used in the training phase. This information is then compared against the baseline statistics that were calculated above. Any port on any host that exhibits activity that differs by more than one standard deviation generates a warning, while alerts are generated if the traffic differs by more than two standard deviations. In order to avoid false alarms from ephemeral port usage, one of two conditions must be met. Either there must be at least 50 flows to the port during the current hour, or the mean for the baseline must have at least 50 flows. This avoids situations where the mean is 2 flows, but 7 flows happened to be observed during the hour under consideration.

IV. Experimental Results

We tested our system on a /16 subnet located in the United States that collected Cisco NetFlow data at the border of their network. The data for this network was divided into two sets: inbound and outbound. All inbound traffic had addresses that were sourced from locations outside of the network and destined for hosts inside of the network. All outbound traffic had source addresses that were sourced from within the network and destined for outside the network.

We established a baseline for each of incoming and outgoing data, using the destination and source ports respectively. As network traffic tends to exhibit diurnal patterns representative of typical workdays, we established separate baselines for four different time periods: weekends, late nights, peak hours, and all other.

We built our baseline on data from the week of January 19 - 25. The first baseline represents weekends and holidays, and uses all of the data from January 24 and 25. We expect traffic to drop to minimal levels during this time and that many of the hosts will not show any activity. The next baseline represents the peak hours of weekday activity, and contains 15:00-19:59 GMT of January 19-24, 2004. We expect to see the maximum number of hosts active during this period and the maximum amount of activity per host. The third baseline is the late night weekday activity, representing hours 05:00-10:59 GMT of 19-24 January, 2004. These traffic levels should only be affected by automated processes and parts of the organization that are running 24 hour operations. The final and fourth baseline file consists of the remaining hours of the weekdays. Host behavior during these periods is expected to be sporadic as people arrive at various hours in the morning, and may leave at various hours during the evening.

In order to generate the baseline files, hourly files containing port counts had to be generated first. These hourly files were then aggregated to generate the baseline files containing the statistical information on each port. For instance, the late night activity baseline consisted of one file for each of the 6 hours in the period. Given that our baselines used 5 weekdays of activity, we needed to create 30 files of intermittent data. The next step calculated the mean and standard deviation of each characterized flow. For instance, host 192.168.1.10 using port 443 may occur a total of 100 times over the course of

ISBN 555555555/$10.00

the 30-hour baseline window. We calculate the mean and standard deviation for this host and port on a per hour basis. These statistics are then recorded into a file that will serve as the baseline for that generic time period. Generating the baseline from the intermittent data takes approximately 1.25 minutes, and generating the intermittent files varies greatly depending on the number of records to be processed. The same process is run for each generic time period baseline.

The results from plotting the peak and late night baselines are illustrated in Figures 1 and 2 respectively. We represent this information in graphical form to allow an administrator to glance quickly at a graph and see if there are any alerts. By presenting this information graphically, rather than just as textual alerts, the administrator might notice patterns of activity, such as groups of hosts demonstrating unusual activity, or persistent unusual activity on the same set of hosts. The graphs represent /16 addresses only, and plot the third octet on the *Y*-axis and the fourth octet on the *X*-axis. Thus each /24 subnet is a horizontal line, with different symbols for servers and workstations. As we expected, there is a significant difference in address space densities between the peak and non-peak hours.

Once the baseline was established for the network, we ran experiments for a day that was not part of the baseline window in order to test the abilities and applicability of the system. One of the experiments was run with peak hour traffic, and one of the experiments was run with late night data. The experiment was performed against both incoming and outgoing data sets. All experiments used data collected on January 29, 2004.

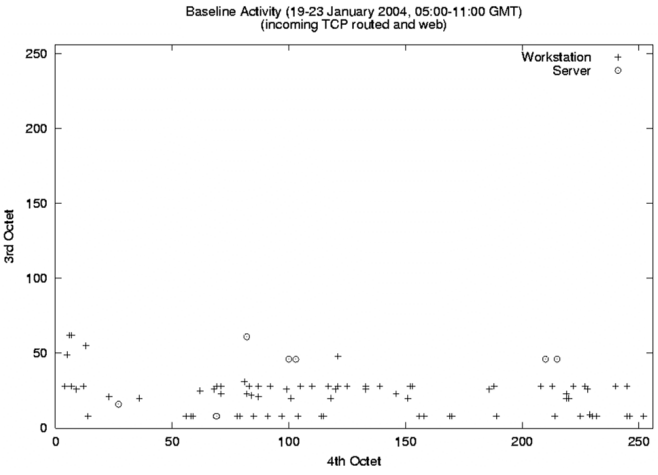

Figure 2. Baseline Activity for Night Hours

In our first experiment, we compared our late night baseline to the hour of data 1000-1100 GMT on January 29, 2004. We illustrate our findings in Figure 3 using the same graphical approach specified earlier, and with warning and alerts using a different shape and color. We also enumerate the program's output in Table 1. As can be seen, there are 5 alerts for 4 IP addresses during this hour. As identified in the section on the training phase, we ignore traffic to all ports that is less than some minimum amount per hour. In this example we used 50 as our cut-off value.

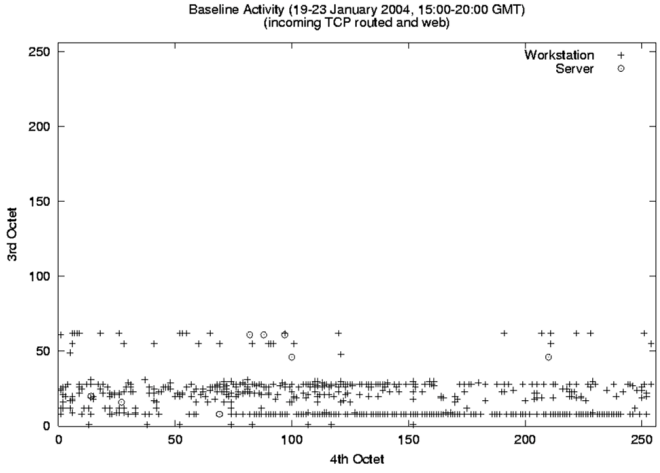

Figure 1. Baseline Activity for Peak Hours

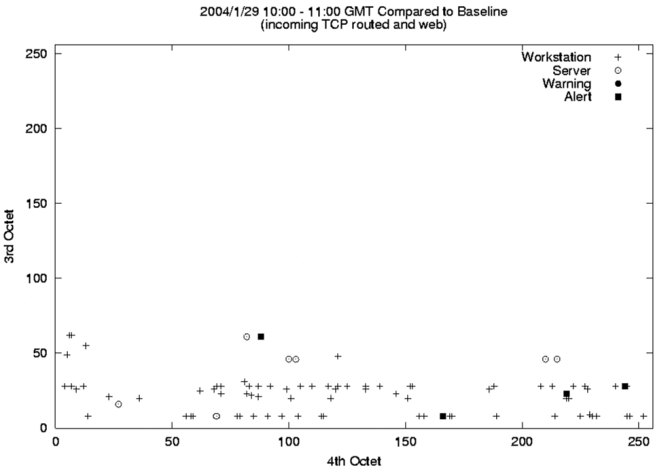

Figure 3. Activity for 1000-1100 GMT January 29, 2004

ISBN 555555555/$10.00

Address	Port	Notification	Actual Measurement	Mean ± Standard Deviation
1.0.8.166	1175	Alert	0	54 ± 11
1.0.8.166	3123	Alert	60	0
1.0.23.219	5101	Alert	53	0
1.0.28.244	3127	Alert	2058	0
1.0.61.88	443	Alert	0	58 ± 59

Table 1. Alert Summary of Activity for 1000-1100 GMT January 29, 2004

The first four alerts involve ports greater than 1024 which would normally be considered ephemeral ports. In the first case, there is no traffic to a particular ephemeral, when it normally sees more than 50 flows per hour on average. It is unusual that an ephemeral port should encounter so much traffic in a single hour, as normally traffic to ephemerals is more distributed. Because of the lasting persistence during the baseline window, this is likely some custom application that was being run during the baseline window. Alternatively, it could have been a back-door system that was in regular use during the baseline period. This highlights one of the limitations of an anomaly detection approach - there is no way to determine normal activity versus malicious other than through more detailed investigations. In either case, the sudden disappearance of the regular traffic on this port likely warrants further investigation.

The next three alerts involve seeing a large amount of traffic directed at ports to which traffic had not previously been observed. Of particular interest is the traffic to port 3127. This particular sample hour was taken during the spread of the MyDoom worm. This shows that this approach can clearly distinguish when there is unusual activity. In this case, the number of flows to that particular port on that particular machine might indicate that it has been compromised. This provides evidence that our approach may be quite powerful in enumerating hosts that are using common new ports to communicate, often indicative of new worm activity.

The last alert was for a Secure Sockets Layer (SSL) server running on port 443 that experiences very sporadic traffic. We have noticed in our testing that some SSL servers have a large variance in their activity. During the 30-hour baseline sample, this port on this host saw between 4 and 198 flows per hour. This variance is indicated in the standard deviation. Therefore, while it is flagged as an alert, it can probably be ignored. Obviously this data point is within one standard deviation; however, it was flagged because it saw no traffic, which was considered unusual. We flag any port that had seen some minimum amount of activity (e.g. 50 flows) in the baseline window, but that did not see activity in the tested window.

ISBN 555555555/$10.00

In our next experiment, we compared our peak hour baseline to the hour of data 1500-1600 GMT January 29, 2004. We illustrate our findings in Figure 4 and enumerate the program's output in Table 2. As can be seen, there are 8 alerts and 3 warnings for 10 unique IP addresses during this hour.

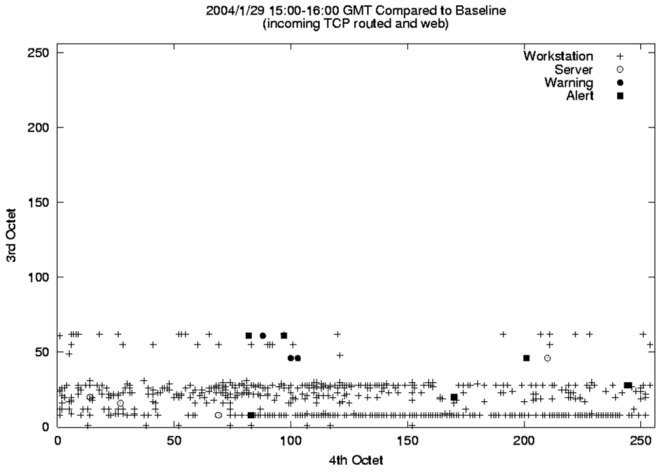

Figure 4. Activity for 1500-1600 GMT January 29, 2004

Table 2. Alert Summary of Activity for 1500-1600 GMT January 29, 2004

Address	Port	Notification	Actual Measurement	Mean ± Standard Deviation
1.0.8.83	1151	Alert	107	0
1.0.20.170	113	Alert	103	0
1.0.28.244	113	Alert	182	0
1.0.28.244	3127	Alert	3127	0
1.0.28.245	113	Alert	56	0
1.0.46.100	443	Warning	2818	1431 ± 716
1.0.46.103	443	Warning	155	97 ± 53
1.0.46.201	80	Alert	0	57 ± 127
1.0.61.82	113	Alert	0	83 ± 60
1.0.61.88	443	Warning	79	599 ± 316
1.0.61.97	443	Alert	0	2014 ± 1942

What is interesting about this is that it shows clear patterns of activity that should be investigated. First, we see the continued large number of connections to port 3127 on 1.0.28.244. This is a strong indication that this host is continuing to demonstrate MyDoom activity. Second, there seems to be a sudden rise in activity

associated with port 113, which should likely be investigated. This sudden appearance of port 113 demonstrates the need for the ability to modify the baseline to account for changing traffic patterns or for missed patterns in the original window. Third, port 443 seems to be seeing unusual traffic, although for the most part this is just a warning and can probably be ignored. A characteristic of SSL servers seems to be this sporadic nature; however, 1.0.61.97 seems to be a popular SSL server that is no longer seeing any traffic. This could be do to an outage and may be cause for further investigation. The administrator would likely ignore the alert on 1.0.46.201 port 80 since the average is only 57 with a very large standard deviation. There is currently no explanation for the traffic on port 1151 for 1.0.8.83.

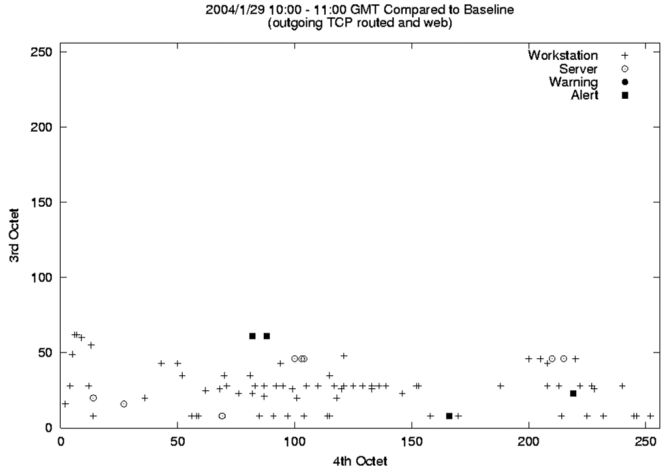

Figure 5. Outbound Activity for 1000-1100 GMT January 29, 2004

Address	Port	Notification	Actual Measurement	Mean ± Standard Deviation
1.0.8.166	1175	Alert	0	51 ± 19
1.0.8.166	3123	Alert	60	0
1.0.23.219	5101	Alert	50	0
1.0.61.82	53709	Alert	82	0
1.0.61.88	443	Alert	208	65 ± 59

Table 3. Alert Summary of Outbound Activity for 1000-1100 GMT January 29, 2004

A means of ensuring that earlier problems are actually occurring and are not due to the fact that we use a unidirectional flow of packets is to run the same approach on the outbound that we did on the inbound traffic. Figure 5 and Table 3 depict the results from the same late night outbound traffic, and Figure 6 and Table 4 depict the results from the same peak hour outbound traffic.

When compared with the traffic in the inbound data set, we see that largely the same machines and ports get flagged. Incoming traffic to 3127 to 1.0.28.244 is missed in the evening, however is suddenly present during the day. This is indicative of a possibly successful compromise of the monitored system. Further investigation would be required to draw more certain conclusions. The host 1.0.61.88 shows lots of activity out on port 443 during the late night test, but there was nothing incoming to the same port and address pair. This type of one-way traffic may be due to some error in the data collection, or might be indicative of a compromise or covert channel. In the peak hours, many of the same machines get flagged as showing unusual activity. Again, we see ephemeral ports broken up across multiple NetFlow records. This might be due to normal activity (in which case we might want to adjust our cut-off from 50 to something higher), or might indicate traffic that should be investigated further.

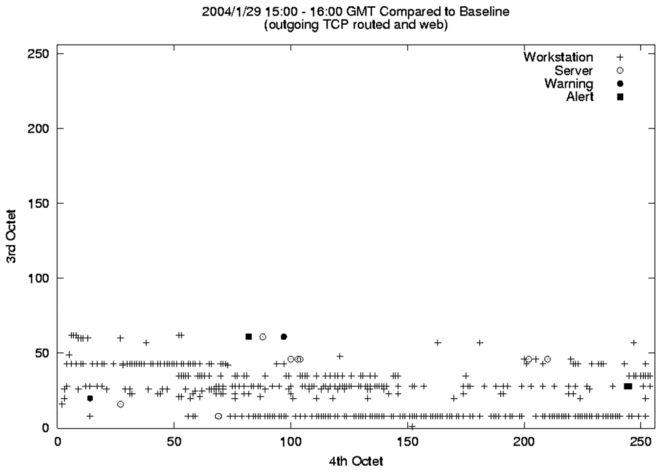

Figure 6. Outbound Activity for 1500-1600 GMT January 29, 2004

Address	Port	Notification	Actual Measurement	Mean ± Standard Deviation
1.0.20.14	25	Warning	129	186 ± 54
1.0.28.244	113	Alert	183	0
1.0.28.244	3127	Alert	2476	0
1.0.28.245	113	Alert	56	0
1.0.61.82	33632	Alert	66	2 ± 1
1.0.61.82	39932	Alert	58	2 ± 2
1.0.61.82	42635	Alert	50	3 ± 4
1.0.61.82	59000	Alert	97	2 ± 3
1.0.61.82	59016	Alert	91	2 ± 3
1.0.61.97	443	Warning	4383	2409 ± 1895

Table 4. Alert Summary of Outbound Activity for 1500-1600 GMT January 29, 2004

ISBN 555555555/$10.00

V. Limitations

Our approach has two known limitations. The first is that we need a better model for recognizing servers. For example, servers seem to appear and disappear between the late night baseline and the peak hour baseline. A possible solution to this is to determine the appropriate values for detecting the presence of a server (perhaps 100 flows/hour on average is too high), or to determine that a machine is a server based on day time traffic and then to maintain that information across baselines. Future work is required to test the effectiveness of other models.

Related to this limitation is the use of standard deviation calculations, which are very sensitive to outliers. As a result, it was not uncommon to see values where the standard deviation was larger than the mean, e.g. a mean of 157 and a standard deviation of 199. One method to address this would be to use quantiles instead of the standard deviation. That is, use a set of data for baselining, calculating both the interquartile range (the values at 25% and at 75%), flagging anything outside this range in yellow, and then using the 2.5% and 97.5% quantiles as the cut-off for flagging anything in red. Additionally, we could use higher order statistics to better describe the distribution; however, a discussion of this is beyond the scope of this paper.

The second issue is the current lack of scalability. This approach does not scale well to networks larger than a class B with the current data structures. First, the performance would decrease rapidly due to heavy usage of memory and disk storage. For instance, a heavily populated class B at peak hours requires 1 Gigabyte of memory to generate the baseline. Additionally, the process is CPU intensive because of the numerous searches needed to match records to the baseline model. The process requires duration on the order of minutes rather than seconds. Further, it is unclear if the visualization of larger networks would be possible using the fairly coarse grained techniques discussed in this paper. The performance of this approach on a class A-sized network or larger is an open question.

VI. Conclusions

This paper contributes a novel approach to detecting the possible compromise of individual hosts by analyzing their behavior at the network level. This approach is based on very simple statistical models, yet experimental results indicate that even these models can yield very useful information. For example, during the peak hour investigated, only five hosts were identified as exhibiting unusual behavior on a /16 subnet. It is believed that even if the system has a high false positive rate, that alerting on such a small number of machines still results in a manageable workload for the security administrator.

We have identified a number of ways that our approach could be improved in the future. First, we would like to match inbound and outbound anomalies and display the results in one report. A simple algorithm could be implemented to sort the list based on some fundamental importance criteria, such as severity of deviation, bi-directional occurrence, and a focus on a port watch list. Second, the server model could be improved by enumerating a list of traditional server ports, which includes a number of ports greater than the 1024 currently implemented. Ideally, we would be able to incorporate domain knowledge about the network rather than rely upon an interpretation of the network traffic to indicate which hosts are servers and which are workstations. Third, we would like to see the inclusion of more sophisticated statistical models that could be used to give a better view of the nature of the anomalies detected.

VII. References

[1] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of LISA '99: 13th Systems Administration Conference*, 1999. Seattle, Washington, USA, November 7-12, 1999.

[2] Vern Paxson. Bro: A system for detecting network intruders in real-time. *In Proceedings of the 7th USENIX Security Symposium*. San Antonio, Texas. January 26-29, 1998.

[3] Levent Ertoz, Eric Eilertson, Aleksandar Lazarevic, Pang-Ning Tan, Paul Dokas, Vipin Kumar and Jaideep Srivastava. Detection of novel network attacks using data mining. In *Proceedings of the 2003 ICDM Workshop on Data Mining for Computer Security*. Melbourne, Florida, USA. November 19, 2003.

[4] D.E. Denning. An intrusion-detection model. IEEE Transactions on Software Engineering. 1987. Vol. 13, no. 2, pages 222-232.

[5] Todd Heberlein, Gihan Dias, Karl Levitt, Biswanath Mukherjee, Jeff Wood, and David Wolber. A network security monitor. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*. Oakland, California, USA. 7-9 May 1990.

[6] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the Third*

ISBN 555555555/$10.00

SIAM Conference on Data Mining. San Francisco, May 2003.

[7] E.E. Eilertson, L. Ertoz and V. Kumar. MINDS: A new approach to the information security process. In *Proceedings of the 24th Army Science Conferece*. November 1999.

[8] C. Leckie and R. Kotagiri. A probabilistic approach to detecting network scans. In *Proceedings of the 2002 IEEE Network Operations and Management Symposium*. Florence, Italy. April 2002.

[9] Hyukjoon Kim, Surrey Kim, Michael A. Kouritzin and Wei Sun. Detecting network portscans through anomaly detection. In *Proceedings of SPIE: Signal Processing, Sensor Fusion, and Target Recognition XIII*. Vol. 5429. Pages 254-263.

[10] M. Iguchi and S. Goto. "Detecting Malicious Activities through Port Profiling." IEICE Transactions on Information and Systems, Volume E82-D, Number 4, April 1999. Pages 784-792.

[11] McAfee Security. http://us.mcafee.com/virusInfo/default.asp?id=helpCenter&hcName=mydoom&cid=9547. Last visited: 16 March 2004.

[12] Carrie Gates, Michael Collins, Michael Duggan, Andrew Kompanek, and Mark Thomas. More NetFlow tools: For performance and security. In *Proceedings of the 18th Large Installation Systems Administration Conference* (LISA 2004). Pages 121 - 132. Atlanta, Georgia. November 14 - 19, 2004.

[13] A. Kompanek and M. Thomas. "SiLK Analysis Suite". http://sourceforge.net/projects/silktools/. Last visited: 16 March 2004.

[14] Cisco Systems. Cisco CNS NetFlow Collection Engine. http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_user_guide_chapter09186a00801ed569.html Last visited: 5 April 2004.

ISBN 555555555/$10.00