Proceedings of the 7th IFIP/IEEE International Symposium on
Integrated Network Management, Seattle, WA - May 14-18, 2001. In Press.

Proactive Detection of Distributed Denial of Service Attacks using MIB Traffic Variables - A Feasibility Study *

João B. D. Cabrera[a], Lundy Lewis[b], Xinzhou Qin[c], Wenke Lee[c], Ravi K. Prasanth[a], B. Ravichandran[a] and Raman K. Mehra[a]

Scientific Systems Company[a]
500 West Cummings Park, Suite 3000
Woburn MA 01801 USA
`cabrera,prasanth,ravi,rkm@ssci.com`

Aprisma Management Technologies[b]
486 Amherst Street
Nashua, NH 03063 USA
`lewis@aprisma.com`

North Carolina State University[c]
Department of Computer Science
Rayleigh, NC 27695 USA
`xqin@unity.ncsu.edu`,
`wenke@csc.ncsu.edu`

Abstract

In this paper we propose a methodology for utilizing Network Management Systems for the early detection of Distributed Denial of Service (DDoS) Attacks. Although there are quite a large number of events that are prior to an attack (e.g. suspicious log-ons, start of processes, addition of new files, sudden shifts in traffic, etc.), in this work we depend solely on information from MIB (Management Information Base) Traffic Variables collected from the systems participating in the Attack. Three types of DDoS attacks were effected on a Research Test Bed, and MIB variables were recorded. Using these datasets, we show how there are indeed MIB-based precursors of DDoS attacks

*This work was supported by the Air Force Research Laboratory (Rome, NY - USA) under contract F30602-00-C-0126 to Scientific Systems Company, and by Aprisma's University Fellowship Program 1999/2000.

that render it possible to detect them before the Target is shut down. Most importantly, we describe how the relevant MIB variables at the Attacker can be extracted automatically using Statistical Tests for Causality. It is shown that Statistical Tests applied in the time series of MIB traffic at the Target and the Attacker are effective in extracting the correct variables for monitoring in the Attacker Machine. Following the extraction of these Key Variables at the Attacker, it is shown that an Anomaly Detection scheme, based on a simple model of the normal rate of change of the key MIBs can be used to determine statistical signatures of attacking behavior. These observations suggest the possibility of an entirely automated procedure centered on Network Management Systems for detecting precursors of Distributed Denial of Service Attacks, and responding to them.

Keywords

Security Management; Data Warehousing and Statistical Methods in Management; Network and Systems Monitoring; Information Modeling

1 Introduction

Security Management (SM) is of growing interest in industry and in research. There are at least four general methods in SM: (i) Scanning, in which knowledge-based software tools looks for security loopholes in a network; (ii) On-line monitoring, in which network management tools are triggered by certain suspicious events; (iii) Data encryption and secure passwords; and (iv) Firewalls. The work described in this paper belongs to the on-line monitoring category, and is therefore closely related to the area of Intrusion Detection (eg. [8], [10]). Our overall objective is to investigate how a Network Management System could be used for Intrusion Detection. In the present effort, we are concerned with Distributed Denial of Service Attacks (DDoS). Our specific goal will be to determine precursors of DDoS attacks so that they can be prevented before causing damage. In the context of Security Management, we view a proactive, or anticipatory action as a procedure consisting of two elements:

1. **Detection** of Security Violations before Network Operation is compromised;

2. **Response** to the attempted Security Violation.

Our discussion in this paper is mainly focussed in the detection element, and the role of Network Management Systems (NMS) - [14]. We expect however that the NMS will also coordinate the response to the attack.

2 Distributed Denial of Service Attacks

Distributed Denial of Service (DDoS) Attacks caused severe disruptions to many corporations in early 2000 ([12]). A technical analysis of these attacks is given in [3]. According to [3], the DDoS attacks have two phases, and involve three classes of systems.

A simplified topology for these attacks is given on Figure 1. Not all of these systems are under supervision from the NMS. The **Master** system coordinates the whole effort.

In the first phase of the attack, the Master infiltrates multiple computer systems, and installs the DDoS tools, which are scripts capable of generating large volumes of traffic under command from the Master. Details on these scripts are given in [3] and references therein. We call these infiltrated systems the **Slaves**. The second phase of the attack cannot take place until phase one is completed.

The second phase is the actual DDoS attack. Under command from the Master, the Slaves generate network traffic to bring down the **Target** system. Any system connected to the Network can be a Target. Routers and Web Servers are typical examples. Although the nature of the traffic (UDP, ICMP, etc.) differs among the various types of DoS attacks (eg. [7]), the common factor is the abnormally large number of connections attempted to the Target system during a very small interval of time ([2], [7]). Although the processing of this traffic usually shuts down the Target system (eg. [7]), many times it does not matter how the Target handles the packets; the volume of traffic is so great that the whole network becomes congested with artificial traffic. The congestion does not allow legitimate traffic to pass, thus rendering the Target unaccessible ([3]), and making the DDoS attack ultimately successful. We assume that the Master is not under NMS monitoring, but the Target and a few Slaves (not all) are. By "being under NMS monitoring" we mean that the NMS is capable of recording the activity of the system. In the case of the Slaves, it does not necessarily mean that the NMS is aware that a particular system is a DDoS Slave; as the attack proceeds however, the NMS may infer it, and act accordingly. We call NMS-Land the ensemble of all systems under the NMS monitoring.

Besides slaves $S4$ and $S5$, all other systems inside NMS-Land along the path between the Master and the Target are places in the Network where significant events can be recorded. These events are observable by the NMS, and precede the ultimate completion of the second phase of the DDoS attacks. **These are the events that we expect to use for proactive detection**. We define two classes of locales[1] in NMS-Land, identified in Figure 1:

1. $M4$ and $M5$ represent the systems inside NMS-Land along the path between the Master and $S4$ and $S5$, respectively.

2. $Gi, i = 1, \cdots, 5$ represent the systems inside NMS-Land along the path between the Slave Si and the Target.

The systems of interest to us are $S4, S5, M4, M5, Gi, i = 1, \cdots, 5$. Figure 2 presents a simplified timeline for the DDoS attacks. We assume that the NMS keeps an universal reference clock. All the times are measured at this clock, irrespective of the NMS being aware or not of the event defining a particular time entry. $T0$ represents the time where the Master starts installing the Slaves. The installation procedure in itself is very complex, as described in [3], and lasts till $T1$. We view $T1$ as the instant when the last

[1]We define a locale as an ensemble of systems, including routers, computers, etc.

Slave was completely installed; $T1$ marks the last communication between the Master and the Slaves before the start of the second phase of the attack. Several recordable events of interest concerning the installation of Slaves during Phase I happen at $S4$ and $S5$, and also at $M4$ and $M5$ during the interval of time between $T0$ and $T1$. At time $T2$ the Master commands the Slaves to initiate the Attack. The decision to start the attack at time $T2$ is made by the Master alone. $T2 - T1$ can be days, or weeks. This is an entirely human decision, made by the Master. The time elapsed between $T2$ and $T5$ is much smaller than $T2 - T1$. Assuming that there is no feedback between Master and Slaves, the sequence of events in the interval $T2$ to $T5$ is entirely determined by the interaction between the DDoS tools residing in the Slaves, and the Network. The systems of interest at this stage are $S4$, $S5$ and $Gi, i = 1, \cdots, 5$.

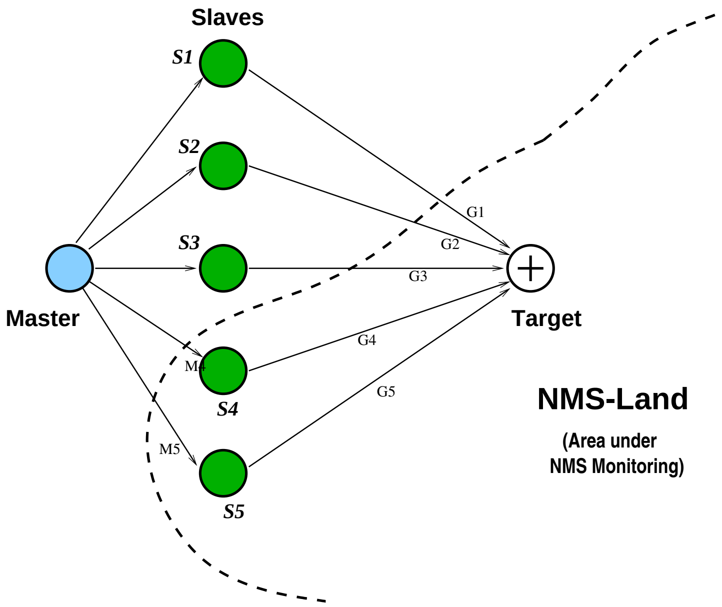

Figure 1: Distributed Denial of Service Attacks - A simplified Topology. The Target and a few Slaves (not all) are under supervision from the NMS in this case.

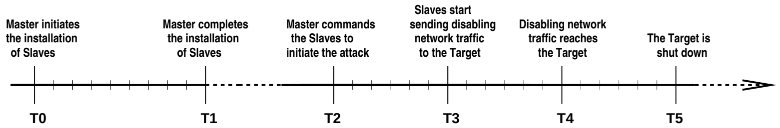

Figure 2: Distributed Denial of Service Attacks - A simplified Timeline.

3 MIB Traffic Variables and DDoS Attacks: Experiments on a Research Test Bed

A Data Set for studying DDoS attacks was produced at North Carolina State University (NCSU) on a research network with the topology depicted on Figure 3. The Network Management System collected 91 MIB variables corresponding to five MIB (Management Information Base) groups[2]: `ip`, `icmp`, `tcp`, `udp` and `snmp`. Variables were collected for intervals of 2 hours, at a sample rate of 5 seconds. The Target OS is Red Hat Linux 6.1 (kernel: 2.2.12-20smp); Attacker 1 OS is Red Hat Linux 6.2 (kernel: 2.2.14-50); Attacker 2 is Sun OS 5.5.1. We used the data corresponding to both Attack Runs and Normal Runs as described below:

- **Attack Runs**: Three types of DDoS attacks produced by TFN2K (Ping Flood and Targa3) and Trin00 (UDP Flood) - see [3] for descriptions of the DDoS tools and attack types. During each of the attacks, MIBs were collected for the Attacking Machine (Attacker 1 or Attacker 2 in Figure 3) and for the Target. The time series for MIB variables corresponding to counter variables were differentiated. Two runs were recorded for each type of attack. According to the terminology introduced in section 2, Attacker 1 and Attacker 2 are Slaves; the Master is not under monitoring from the Network Management System.

- **Normal Runs**: MIBs were collected during times when the machines were not being targets of attacks, nor being the attackers. 12 runs are available for the Target Machine, 7 runs are available for Attacker 1, and 14 runs are available for Attacker 2.

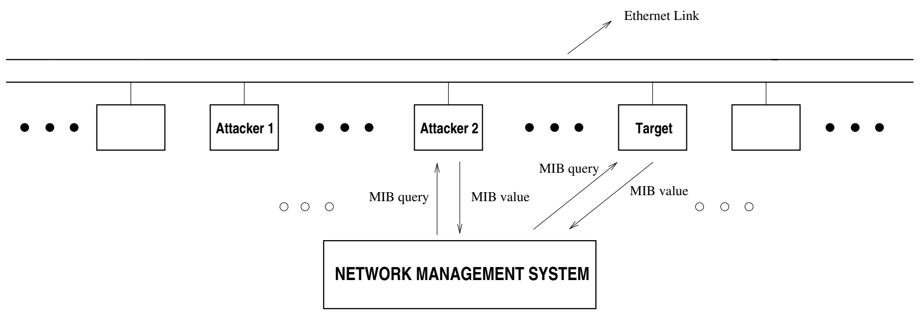

Figure 3: The research network at NCSU.

The data set includes events starting on T2; the DDoS tools are assumed to be already installed in the Attacking Machines when the Attack Runs start. Hence, prospective

[2]Definitions of the MIB variables for all groups are available in [13], pp. 363-388.

Proactive Rules should relate events in T2 or T3 with events in T4 and T5. To illustrate the nature of the MIB variables and their relevance for attack detection, Figure 4 depicts `icmpInEchos` at the Target, aligned with four MIB variables at the Attacker Machine that show remarkable activity before the pings reach the target. These are `ipOutRequests`, `icmpInEchoReps`, `tcpInErrs` and `udpInErrors`. Three remarks can be made:

1. Although the outbound pings are not visible at the ICMP level, it appears that there is a large number of IP requests at the Attacker preceding the flood at the Target. This would be a T3 event, according to Figure 2.
2. About 30 samples before the first Ping flood, and about 60 samples before the second ping flood, the Attacker receives a few ICMPECHOREPLY packets. According to [3], the communication between Master and Slave in TFN2K happens through ICMP, UDP or TCP. These ICMPECHOREPLY packets are the command from the Master to the Slave to initiate the attack, i.e. these constitute a T2 event, according to Figure 2.
3. Two other MIBs at the TCP and UDP groups in the Attacker also show variations that coincide with the variations in `icmpInEchoReps`.

These four variables were obtained from domain knowledge about the TFN2K Ping Flood attack. In practice, we need a procedure to extract Key Variables for the Attacker automatically, from the entire collection of MIB data at the Attacker Machine. Such a procedure is described in section 4, and the results are presented for the case of the TFN2K Ping Flood Attack, the TFN2K Targa3 attack, and for the Trin00 UDP Flood Attack.

4 Automated Proactive Detection of DDoS Attacks: A Methodology and Experimental Results

Given a large database describing the operation of an Information System, we view the problem of extracting Proactive Rules for Security as consisting of the three steps delineated below. These steps are performed off-line, and produce a set of rules to be used for detecting security violations on-line. While our focus in this paper are in Distributed Denial-of-Service Attacks, we expect that such methodology can be applied to other types of Security Violations, in which causal relationships could be inferred from measured variables.

1. **Detecting Attacks**. The objective here is to determine which are the variables in the Target Machine that better characterize the occurrence of an attack (events T4 and T5). We call these variables the Key Variables at the Target. Recall that this step is performed off-line, akin to a forensic examination; one already knows that an attack took place. The final product of this step is the list of Key Variables

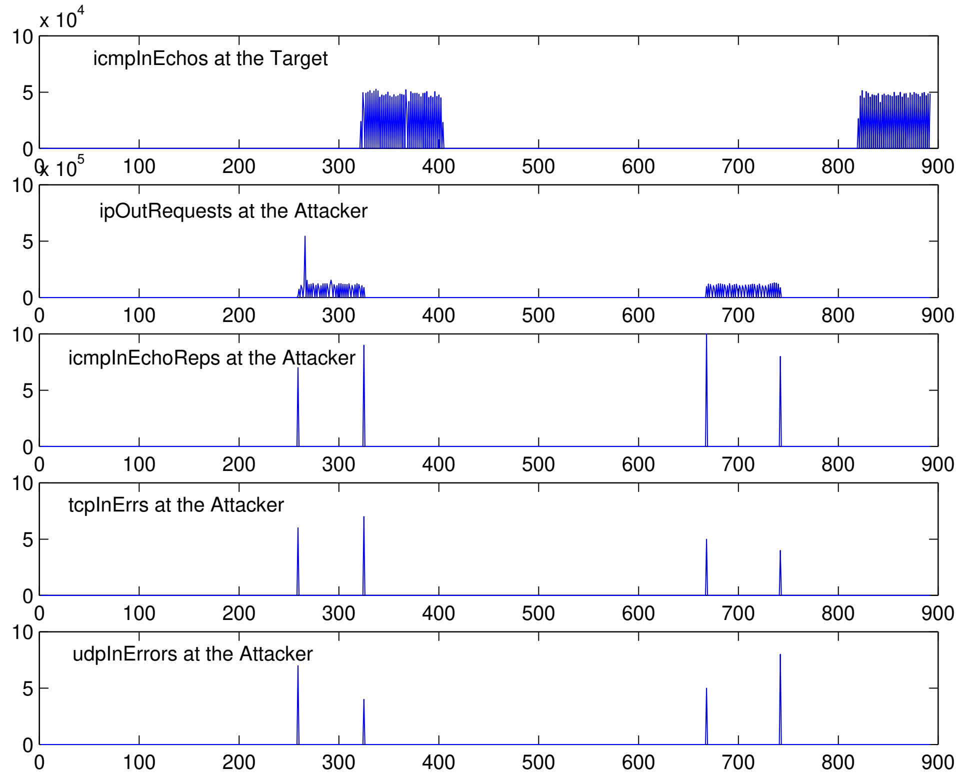

Figure 4: TFN2K Ping Flood: Selected MIB variables at the Attacker and at the Target.

at the Target. There are two procedures for determining the Key Variables at the Target. One way is to use domain knowledge about the attack. For example, for the Ping Flood illustrated on Figure 4, it is *known* that `icmpInEchos` is the right variable to look, since Ping Floods are effected by sending `icmpInEchos` packets to a Target. A second way is to compare the evolution of each variable during an attack with the evolution of the variable during normal operation. Variables that display a large variation between normal operation and attack, should be declared Key Variables at the Target. Since we are looking to localized variations in the variables, the time series should be segmented on small sub-time series, which are then compared with normal profiles. This procedure was used in [15] and [16] for detecting anomalies in network operation, due to component faults. Anomalies were detected as variations on the parameters of AutoRegressive models. For the case of Denial-of-Service attacks however, the traffic variations in the Target are so intense, that much simpler procedures can be employed. The situation in Figure 4 is typical; `icmpInEchos` grows from 0 to $50,000$ abruptly. Just by averaging the time series along properly chosen intervals, it is possible to detect the presence of such attacks. On [2] it is shown that for large classes of Denial-of-Service attacks, the traffic variables represent the main feature of interest. In this paper, we utilized domain knowledge about the attacks for extracting the Key Variables at the Target.

2. **Detecting Correlations**. Once the Key Variables at the Target are determined, we need to determine (if possible) variables in the prospective Attacker Machines that are causally related with them. These variables at Attacker Machines are related to events T2 and T3. Recall that we do not know which ones are the Attacking Machines; we only know a list of candidates, and their corresponding variables. We make the **assumption** that any causal relationship between variables at prospective Attackers and the Key Variables at the Target is to be inferred as a link between the Attacker and the Target. This assumption need to be verified. The final product of this step is the list of Key Variables at the Attacker.

3. **Detecting Precursors to Attacks**. Following the detection of Correlations, the objective here is to determine particular features at the Key Variables at the Attacker that precede the attack at the Target. Recall that these variables were found in Step 2 to be causally related with the Attack; hence we may **expect** that certain anomalies in these variables may be indicative of an incoming attack. Once these features are determined, and are shown to precede the Attack, we can construct Proactive Rules, that constitute the end product of Step 3. These Proactive Rules can be used for implementing alarms on an NMS.

These three components are detailed in the sequel.

4.1 Step 1: Detecting Attacks

Inspection of the DDoS scripts and experiments led us to conclude that the Key Variables at the Target for the three attacks are:

1. **TFN2K Ping Flood** The Ping Flood attack is effected by sending a large amount of ICMPECHOREQUEST packets to the Target. Clearly, `icmpInEchos` is the Key Variable at the Target in this case.

2. **TFN2K Targa3** The Targa 3 attack is effected by sending combinations of uncommon IP packets to the Target. These uncommon packets consist of invalid fragmentation, protocol, packet size, header values, options, offsets, TCP segments and routing flags. MIB variables reflecting errors at different layers cab be used as Key Variables at the Target. We selected `ipReasmFails` in our experiments.

3. **Trin00 UDP Flood** the UDP Flood Denial-of-Service Attack is created when the Attacker sends UDP packets to random ports on the Target. We utilized `udpInDatagrams` as the Key Variable at the Target in this case.

4.2 Step 2: Detecting Correlations

In this step, we attempt to determine the variables at the Attacker Machine that are causally related to the Key Variables in the Target. Since we are looking for Proactive Rules, we need to determine variables at the Attacker that contain events that precede the Attack detection at the Target. These events can be T2 events, or T3 events, as described in section 2. In this section, we investigate the use of Causality Tests (eg. [5], [6]), for determining these Key Variables at the Attacker. Testing for causality in the sense of Granger, involves using statistical tools for testing whether *lagged* information on a variable u provides any statistically significant information about a variable y. If not, then u does not Granger-cause y. Following [6], the Granger Causality Test (GCT) compares the residuals of an AutoRegressive Model (AR Model) with the residuals of an AutoRegressive Moving Average Model (ARMA Model). Assume a particular lag length p, and estimate the following unrestricted equation:

$$y(k+1) = \sum_{i=1}^{p} \alpha_i y(k-i+1) + \sum_{i=1}^{p} \beta_i u(k-i+1) + e_1(k)$$

The Null Hypothesis H_0 of the GCT is given by:

$$H_0: \qquad \beta_i = 0, \;\; i = 1, 2, \cdots, p,$$

i.e. u does not affect y up to a delay of p units. The null hypothesis is tested by estimating the parameters of the following restricted equation

$$y(k+1) = \sum_{i=1}^{p} \gamma_i y(k-i+1) + e_0(k)$$

Let R_1 and R_0 denote the sum of the squared residuals under the two cases:

$$R_1 = \sum_{t=1}^{T} e_1^2(t), \qquad R_0 = \sum_{t=1}^{T} e_0^2(t)$$

If the test statistic g given by:

$$g = \frac{(R_0 - R_1)/p}{R_1/(T-2p-1)} \sim F(p, T-2p-1)$$

is greater than the specified critical value, then reject the null hypothesis that u does not Granger-cause y. Here, $F(a, b)$ is Fisher's F distribution with parameters a and b ([6]). In other words, high values of g are to be understood as representing strong evidence that u is causally related to y. In the traditional sense, we say that u_1 is more likely to u_2 to be causally related with y if $g_1 > g_2$, where $g_i, i = 1, 2$ denote the GCT statistic for the input-output pair (u_i, y).

4.2.1 Using the GCT for finding causal relations on a DDoS Attack

We apply the GCT, by comparing the residuals of the AR Model corresponding to the Key Variables at the Target, with the ARMA Models corresponding to the input-output pairs where the output is the Key Variable at the Target, and the inputs are one of the 64 MIB variables at the Attacker, corresponding to the `ip`, `icmp`, `tcp` and `udp` groups. Based on [3], we have domain knowledge (ground truth) about the Key Variables at the Attacker. Tables 1 and 2 display these variables. We have used the Deterministic Stochastic Realization Algorithm (DSRA) to fit ARMA models to input-output pairs, and the Stochastic Realization Algorithm (SRA) to fit AR models to the output ([9], [11]). The GCT was applied for two runs of each of the three types of DDoS attacks. T4 events happen more than once in each run; for example, as shown in Figure 4, the Target Machine receives two "volleys" of pings during the course of Run 1 for TFN2K Ping Flood. The duration of the runs also varies from case to case. Table 3 gives the relevant information for each of the Runs.

MIB	Event
`icmpInEchoReps`	T2
`tcpInErrs`	T2
`tcpInSegs`	T2
`udpInErrors`	T2
`udpInDatagrams`	T2
`ipOutRequests`	T3

Table 1: Key Variables at the Attacker for TFN2K - Ground Truth.

MIB	Event
`udpInDatagrams`	T2
`udpOutDatagrams`	T3
`ipOutRequests`	T3

Table 2: Key Variables at the Attacker for Trin00 - Ground Truth.

DDoS Attack	Run	Samples	T4 Ev.	p	T	99%	95%
TFN2K Ping Flood	1	892	2	100	792	1.40	1.27
TFN2K Ping Flood	2	1016	3	120	896	1.37	1.24
TFN2K Targa3	1	825	3	100	725	1.41	1.27
TFN2K Targa3	2	977	3	100	877	1.40	1.27
Trin00 UDP Flood	1	582	2	80	502	1.47	1.31
Trin00 UDP Flood	2	991	3	100	891	1.39	1.27

Table 3: Statistics for the Attack Runs, parameters for the GCT, and thresholds for the g statistic for two significance levels. These thresholds are the critical levels corresponding to $F(p, T-2p-1)$.

4.2.2 The Results

We consider a scenario in which there are nine potential Attackers against the Target: the true attacker and eight decoys corresponding to the normal runs. We then apply the GCT to measure the causality strength of all MIB variables in the potential attackers, with respect to the Key Variable at the Target in each of the Attacks. MIB variables at potential attackers resulting on a GCT statistic above the threshold for 95% significance level were considered to Granger-cause the Key Variables at the Target, and where kept for analysis in Step 3. We count detections whenever the ground-truth variables presented in Table 1 and 2 are correctly picked by the GCT. False alarms correspond to MIB variables being flagged in the decoys. As an example, Tables 4 and 5 display the results for the first run of the TFN2K Ping Flood Attack. Table 6 summarizes the results for the three types of attacks. Notice that in Run 1 for the TFN2K Ping Flood attack, besides the six "true" MIB variables, the GCT also detected other five MIB variables at the Attacker Machine. These are related to the "true" MIB variables through Case Diagrams (eg. [13]), and are also causally related to the Key Variable at the Target. The same observation also applies for Run 2 of the TFN2K Ping Flood attack, and to other runs of other attacks. Concerning Table 6, Notice that at least one "true" MIB variable at the Attacker is detected in each run. This is all one needs to set up an alarm for Proactive Detection. The FA (False Alarm) Rate for Decoy MIBs is obtained by computing the total number of significant MIB variables found in all normal runs, divided by the total number of MIB variables. For example, in Run 1 for Ping 1, Table 5 indicates 23 out of 64×8 , giving the FA rate of 4.49% recorded in Table 6.

Rank	MIB	g
1	`ipOutRequests` (T3)	5.26
2	`tcpInErrs` (T2)	3.50
3	`ipInReceives`	2.67
4	`ipInDelivers`	2.65
5	`udpInErrs` (T2)	2.63
6	`udpOutDatagrams`	2.58
7	`udpInDatagrams` (T2)	2.57
8	`icmpInEchoReps` (T2)	2.04
9	`icmpInMsgs`	1.99
10	`tcpInSegs` (T2)	1.31
11	`udpNoPorts`	1.27

Table 4: TFN2K Ping Flood Run 1: Top MIBs at the Attacker according to the g statistic.

Run	Max g	No. Sign. MIBs
1	2.41	7
2	0.77	0
3	0.64	0
4	1.62	3
5	1.79	1
6	3.25	10
7	1.29	2
8	1.08	0

Table 5: TFN2K Ping Flood Run 1: Performance of the GCT for normal runs at the Attacker. The significance level is 95%.

DDoS Attack	Run	Detections	FA per Decoy MIBs (%)
TFN2K Ping Flood	1	6/6	4.49
TFN2K Ping Flood	2	1/6	3.13
TFN2K Targa3	1	3/6	3.71
TFN2K Targa3	2	1/6	2.68
Trin00 UDP Flood	1	3/3	5.08
Trin00 UDP Flood	2	3/3	7.59

Table 6: Results of Step 2: Detection Rates and FA Rates for MIB variables that contain precursors to DDoS Attacks.

4.3 Step 3: Detecting Precursors to Attacks

The Key Variables at the Attacker determined in Step 2 are labeled as causally related with the Attack at the Target, but we still need to find a trigger, or a Key Event at the Attacker. This is an Anomaly Detection Problem; we postulate that any anomalous behavior at Key Variables at the Attacker are to be considered Key Events at the Attacker. One possible approach is to look for jumps in the MIB variables, by monitoring the absolute values of the differentiated time series $z(k) = |y(k) - y(k-1)|$. Using 12 Normal Runs, we constructed a *Normal Profile of Jumps* for each of the 64 MIB variables. Given a Key Attacker Variable determined on Step 2, Key Events at the Attacker are defined as jumps larger than the largest jump encountered the *Normal Profile of Jumps*. Key Attacker Variables with no Key Events are discarded. The Key Events are used to set the alarms.

4.3.1 The Results

As shown in Table 7, We have found that this procedure led to a substantial reduction of the False Alarms produced on Step 2, with small reductions in the detection rates. Notice that we are still detecting at least one valid precursor at each Attack Run. These results suggest that Step 3 serves to effectively prune an initial ensemble of candidate MIB variables determined used GCT, and produce a final ensemble with more manageable FA rates.

Remark 1 We have verified that the maximum jumps occurring in the Key Variables at the Attacker precedes the Attack at the Target in all cases, signifying that our methodology actually extracted the right events for enabling Proactive Detection.

Remark 2 The significance levels on GCT can be used as a tuning parameter for effecting a trade-off between detections and FAs at Step 2. Larger significance levels lead to higher thresholds, and vice-versa.

DDoS Attack	Run	Detections	FA per Decoy MIBs (%)
TFN2K Ping Flood	1	4/6	1.37
TFN2K Ping Flood	2	1/6	0.52
TFN2K Targa3	1	1/6	0.78
TFN2K Targa3	2	1/6	0.22
Trin00 UDP Flood	1	2/3	0.98
Trin00 UDP Flood	2	1/3	1.17

Table 7: Final Results: Detection Rates and FA Rates for Events at MIB variables.

5 Conclusions and Further Work

In this paper we discussed a methodology for automatically extracting probable precursors of DDoS attacks using MIB Traffic Variables. For all three attacks under study, the methodology extracts at least one valid attack precursor, with rates of false alarm of about 1%. Since the framework depends on MIB information alone, it is straightforward to use these Statistical signatures to implement MIB watches in common Network Management Systems. While we found these results to be encouraging, we are well aware that these were limited experiments, on a local test bed, under controlled traffic loads. Further work is needed to validate our methodology under more realistic traffic conditions. To be applicable in a realistic setting, multiple domains are likely to be involved. An attack might commence in Seattle, while the target is in Miami. Thus, our plans for further work are to scale the approach to multi-domain environments. Two problems arise at once: (i) network management systems in multiple domains will have to be in communication (e.g. [1], [4]) and (ii) an "impending attack" alert and offending packets might find themselves racing towards the same domain, where the former's destination is the target and the latter's destination is a management system for the target. Possible solutions to the racing problem come to mind immediately; for example, the alert might be sent via phone, pager, or satellite.

Acknowledgment Scientific Systems Company acknowledges the continuing support from the Defensive Information Warfare Branch at the Air Force Research Laboratory in Rome, NY. We are particularly grateful to Peter J. Radesi from AFRL, for his encouragement during the course of this effort.

References

[1] K. Boudaoud, H. Labiod, R. Boutaba, and Z. Guessoum. Network Security Management with Intelligent Agents. *Proceedings of NOMS*, 2000. IEEE Publishing.

[2] J. B. D. Cabrera, B. Ravichandran, and R. K. Mehra. Statistical Traffic Modeling for Network Intrusion Detection. In *Proceedings of the Eighth International*

Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, pages 466–473, San Francisco, CA, August 2000. IEEE Computer Society.

[3] P. J. Criscuolo. Distributed Denial of Service - Trin00, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht. Technical Report CIAC-2319, Department of Energy - CIAC (Computer Incident Advisory Capability), February 2000.

[4] Z. Fu, H. Huang, T. Wu, S. Wu, F. Gong, C. Xu, and I. Baldine. ISCP: Design and Implementation of an Inter-Domain Security Management Agent (SMA) Coordination Protocol. *Proceedings of NOMS*, 2000. IEEE Publishing.

[5] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 34:424–438, 1969.

[6] J. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[7] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, Massachusetts Institute of Technology, June 1999.

[8] W. Lee, S. J. Stolfo, and K. W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

[9] R. K. Mehra, K. M. Nagpal, and R. K. Prasanth. Deterministic-stochastic realization algorithms (DSRA). Technical report, Scientific Systems Company, Inc., 1997.

[10] B. Mukherjee, L. T. Heberlein, and K. N. Levitt. Network intrusion detection. *IEEE Network*, 8(3):26–41, May 1994.

[11] P. van Overschee and B. De Moor. *Subspace Identification for Linear Systems*. Kluwer Academic Publishers, Norwell, MA, 1996.

[12] Science and Technology Section. Internet Security - Anatomy of an Attack. *The Economist*, February 19, 2000, pages 80-81.

[13] W. R. Stevens. *TCP/IP Illustrated, Vol. 1: The Protocols*. Addison-Wesley, 1994.

[14] M. Subramanian. *Network Management - Principles and Practice*. Addison-Wesley, 2000.

[15] M. Thottan and C. Ji. Proactive anomaly detection using distributed agents. *IEEE Network*, pages 21–27, September 1998.

[16] F. Zhang and J. Hellerstein. An Approach to On-Line Predictive Detection. In *Proceedings of the Eighth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, pages 549–556, San Francisco, CA, August 2000. IEEE Computer Society.