Journal of Network and Systems Management, Vol. 10, No. 2, June 2002 (© 2002)

Proactive Intrusion Detection and Distributed Denial of Service Attacks—A Case Study in Security Management[1]

João B. D. Cabrera,[2,5] Lundy Lewis,[3] Xinzhou Qin,[4] Wenke Lee,[4] and Raman K. Mehra[2]

Little or no integration exists today between Intrusion Detection Systems (IDSs) and SNMP-based Network Management Systems (NMSs), in spite of the extensive monitoring and alarming capabilities offered by commercial NMSs. This difficulty is mainly associated with the distinct data sources used by the two systems: packet traffic and audit records for IDSs and SNMP MIB variables for NMSs. In this paper we propose and evaluate a methodology for utilizing NMSs for the early detection of Distributed Denial of Service attacks (DDoS). A principled approach is described for discovering precursors to DDoS attacks in databases formed by MIB variables recorded from multiple domains in networked information systems. The approach is rooted in time series quantization, and in the application of the Granger Causality Test of classical statistics for selecting variables that are likely to contain precursors. A methodology is proposed for discovering precursor rules from databases containing time series related to different regimes of a system. These precursor rules relate precursor events extracted from input time series with phenomenon events extracted from output time series. Using MIB datasets collected from real experiments involving Distributed Denial of Service Attacks, it is shown that precursor rules relating activities at attacking machines with traffic floods at target machines can be extracted by the methodology. The technology has extensive applications for security management: it enables security analysts to better understand the evolution of complex computer attacks, it can be used

[1] A preliminary version of this work appeared in the *Proceedings of the Seventh IFIP/IEEE International Symposium on Integrated Network Management*, Seattle, Washington, May 2001.

[2] Scientific Systems Company, 500 West Cummings Park, Suite 3000, Woburn, Massachusetts 01801. E-mail: {*cabrera;rkm*}*@ssci.com*

[3] Aprisma Management Technologies, 121 Technology Drive, Durham, New Hampshire 03824. E-mail: *lewis@aprisma.com*

[4] Georgia Institute of Technology, College of Computing, 801 Atlantic Drive, Atlanta, Georgia 30332. E-mail: *xinzhou;wenke@cc.gatech.edu*

[5] To whom correspondence should be addressed.

1064-7570/02/0600-0225/0

to trigger alarms indicating that an attack is imminent, or it can be used to reduce the false alarm rates of conventional IDSs.

KEY WORDS: Security management; data warehousing and statistical methods in management; network and systems monitoring; information modeling; intrusion detection; distributed denial of service attacks.

1. INTRODUCTION

The failure of authenticators and other protection mechanisms to provide adequate defense to attacks against information systems, and the resulting mistrust in these mechanisms are the most important driving forces behind the development of Intrusion Detection Systems (IDSs) in the past twenty years [1,2]. Current IDSs are not preventive security measures, and are most often used in conjunction with protection mechanisms, such as firewalls, smart cards, and virtual private networks [3]. The combination of several disparate mechanisms into a common security architecture constitutes the defense-in-depth paradigm discussed in detail [4], which is currently in vogue. Allen *et al.* [5] noted that one of the current gaps in the development of IDSs are their inability to interact with other networking elements.

SNMP-based Network Management Systems are well known, and viable in industry. In the early 1990s, these systems were designed following the classic FCAPS (Fault, Configuration, Accounting, Performance and Security), management model defined by the International Standard Organization. According to the FCAPS model, security management controls access to and protects both the network and network management systems against intentional or accidental abuse, unauthorized access, and communication loss [6]. According to this definition, the deployment of IDSs belongs to the realm of security management, and it comes a surprise that IDSs have not been designed to take advantage of the monitoring and alarming infrastructure provided by commercial NMSs. One of the main reasons for this difficulty is the semantic disparity between current IDSs and NMSs. IDSs either rely on audit records collected from hosts (host-based IDSs) or on raw packet traffic collected from the communication medium (network-based IDSs) [7]. SNMP-based NMSs on the other hand rely on MIB variables to set traps and perform polling [8]. In summary, IDSs observe and understand the environment in terms of audit records and packets, while NMSs observe and understand the environment in terms of MIB variables. Returning to Allen *et al*. [5], the fact that some IDSs are able to *communicate* with Network Management Systems via SNMP does not alleviate the problem [5]; the key issue is appropriate semantic interaction among disparate systems, rather than the control of individual IDSs. Clearly, if IDSs could be built using variables that could be understood by NMSs—the MIB variables, then the whole infrastructure already provided by commercially available NMSs could be used to ease the integration of IDSs into the fabric of security management. It is

clear however that not all security violations are amenable to be described in terms of MIB variables, and therefore such "NMS-friendly" IDSs can only provide a partial solution to the problem. On the other hand, MIB variables are yet one more source of data regarding the network infrastructure, and may bring information which would not be available in audit records and packet traffic. Hence, if on one hand "NMS-friendly" IDSs may not able to cover the whole spectrum of security violations, on other hand they may allow the detection of security violations that may not have been detected otherwise. In this paper, and in other related papers by Qin *et al.* [9,10]—we are addressing these issues. Our overall objective is to integrate IDSs with NMSs by attempting to construct IDSs based on MIB variables, or in other data formats that can be easily integrated with NMSs, such as SNMP traps and alarm outputs.

In the course of our research, we have realized that MIB variables can be used not only to characterize security violations, but also to characterize *precursors* to security violations. It led us to the concept of Proactive Intrusion Detection and its integration with security management, which are the main subjects of this paper.

The paper is organized as follows: in Section 2 we introduce the concept of Proactive Intrusion Detection. We attempt to present the general ideas, contrasting Proactive Intrusion Detection with classical Intrusion Detection. Section 3 discusses Distributed Denial of Service Attacks, and their characterization through MIB variables. Section 4 describes the experiments performed to collect data sets related to three types of Denial of Service Attacks. Section 5 presents a methodology for designing Proactive Intrusion Detectors using MIB variables. Section 6 presents the results obtained while applying this methodology to the datasets described in Section 4. Section 7 describes the possible roles that can be played by Proactive Intrusion Detection in security management. Section 8 closes the paper, with our conclusions.

2. PROACTIVE INTRUSION DETECTION

The rules produced by current IDSs are passive in the sense that a security violation has to occur in order to be detected. If we are fortunate to have detection happening early enough, it may be possible to minimize, or even eliminate the deleterious effects of the security violation. But early detection in current IDSs is usually the result of incidental circumstances, not of systematic design. On the other hand, almost all security violations encountered in practice evolve in multiple stages, and some of the preliminary stages may not be destructive *per se*, but merely preparatory steps in the attack scenario. If one could detect indicators of these preparatory steps, or attack precursors, and take immediate action, the resulting attack would be prevented. We call this capability Proactive, or Anticipatory Intrusion Detection, to distinguish it from the passive detection enabled by current IDSs. If successful, Proactive Intrusion Detection would be an invaluable enabling

capability for response, since enough time would be available to respond to the destructive phase of the attack, ideally preventing it to ever take place.

A proactive, or anticipatory intrusion detector is a scheme that issues alarms based on Temporal Rules. For a chapter length discussion see Cabrera *et al.* [11]. In contrast with Passive Intrusion Detectors, Proactive Intrusion Detectors are characterized by:

- *Temporal rules* The antecedent and the consequent items of the detection rule occur at distinct instants of time; the antecedent precedes the consequent. The design of a proactive IDS is a problem in time series analysis [12,13].
- *Report incoming danger* If the antecedent is true, a security violation occurs within a certain amount of time.

The extraction of temporal rules is performed off-line. Large data sets recorded from the multiple domains of the information system are analyzed, in search of rules relating security violations at the target with variables that *may* contain precursors. These data sets include attack runs, in which attacks were known to be present, and also normal, or attack-free runs, used to construct profiles of normal operation. Figure 1 depicts the proposed scheme. The Temporal Correlation

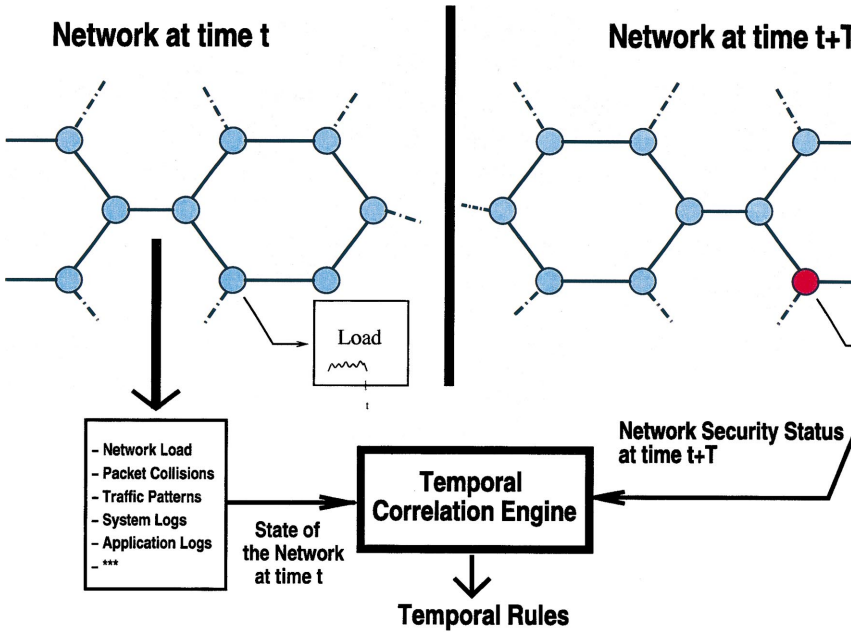


Fig. 1. Proactive Intrusion Detection—Extracting Temporal Rules.

Engine is presented with the evolution of the network states, and the history of security violations. The engine extracts temporal rules, relating network states that precede security violations. The design of a proactive intrusion detector follow four steps, which will be formalized and examined in detail in Section 5:

- *Step 1*: Determining the variables at the targets that characterize the security violation. This is performed by using domain knowledge about the type of attack, or by data-based methods. This step essentially corresponds to the design of a passive IDS.
- *Step 2*: Determining key variables containing attack precursors. This is effected through Statistical Causality Tests, that measure the relative causality strengths between a number of candidate variables and the variables determined in Step 1.
- *Step 3*: Determining the events most likely to be the ones preceding the security violation, or the precursor events. This is performed by comparing the evolution of the variables determined in Step 2 with their normal profile. Deviations from the profile characterize the precursors we are seeking.
- *Step 4*: Verifying if the precursor events extracted in Step 3 are consistently followed by the security violations observed in the variables determined in Step 1.

3. DISTRIBUTED DENIAL OF SERVICE ATTACKS

Distributed Denial of Service (DDoS) attacks have been receiving large attention from the general media since early 2000 when a series of massive Denial of Service attacks incapacitated several popular e-commerce sites [14]. Moore *et al.* [15] described a technique called backscatter analysis to estimate the prevalence of DoS attacks in the Internet. During a three week period on February 2001 over 12,000 DoS attacks were observed, on over 5,000 distinct Internet hosts belonging to more than 2,000 distinct organizations. A technical analysis of DDoS attacks is given by Criscuolo [16]. According to Criscuolo [16], the DDoS attacks have two phases, and involve three classes of systems. A simplified topology for these attacks is given on Fig. 2. 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 by Criscuolo [16] 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

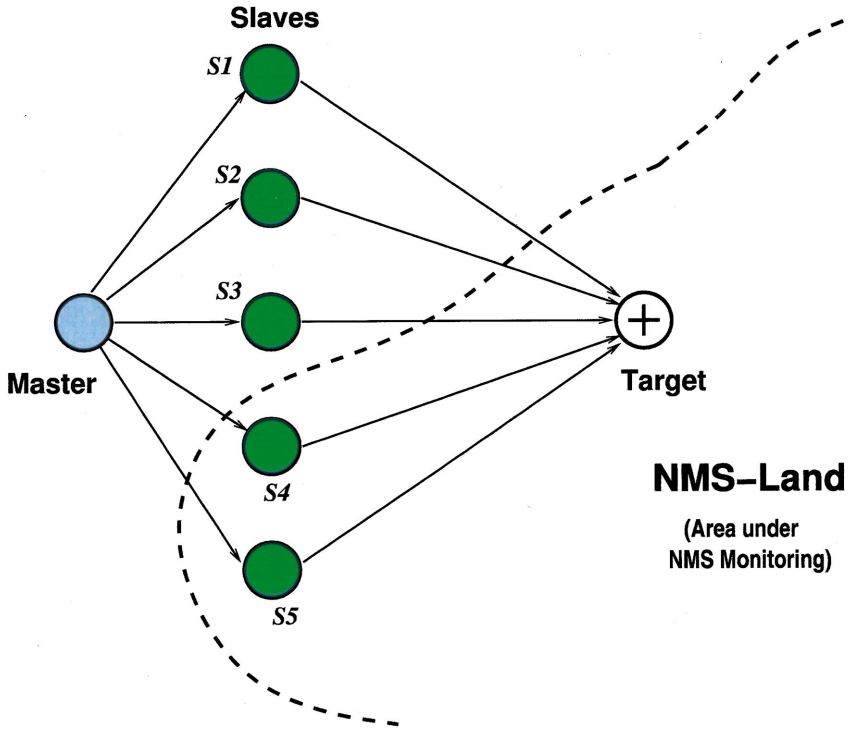


Fig. 2. Distributed Denial of Service Attacks—A simplified Topology. The Target and a few Slaves (not all) are under supervision from the NMS in this case.

examples. Although the nature of the traffic (UDP, ICMP, etc.) differs among the various types of DoS attacks (e.g., Kendall [17]), the common factor is the abnormally large number of connections attempted to the Target system during a very small interval of time [17,18]. Although the processing of this traffic usually shuts down the target system (e.g., Kendall [17]), 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 [16], 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.

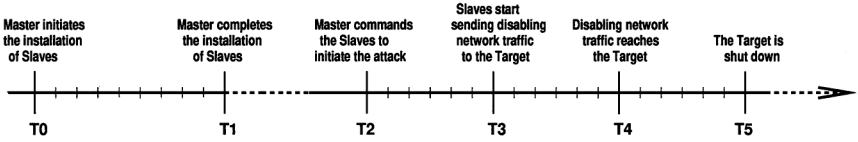


Fig. 3. Distributed Denial of Service Attacks—A simplified Timeline.

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. Figure 3 presents a simplified timeline for the DDoS attacks. $T0$ represents the time where the master starts installing the slaves. The installation procedure in itself is very complex, as described by Criscuolo [16], and lasts till $T1$. We view $T1$ as the instant when the last 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 $S5$ 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.

Now, having described the general nature of DDoS attacks, we proceed to set up experiments in which (i) DDoS attacks are simulated; (ii) MIB data is collected from all systems involved; and (iii) the data is analyzed with statistical algorithms in order to learn probable precursors of target shut-downs. The reader should keep in mind that we want to see how far prior to $T5$ (Fig. 3) we might be able to predict an impending shut-down at $T5$.

4. 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)[6] on a research network with the topology depicted on Fig. 4. The Network Management System collected 91 MIB variables corresponding to five MIB groups[7]: `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);

[6] Authors Qin and Lee were at NCSU prior to August 2001.
[7] Definitions of the MIB variables for all groups are available see Stevens [19].

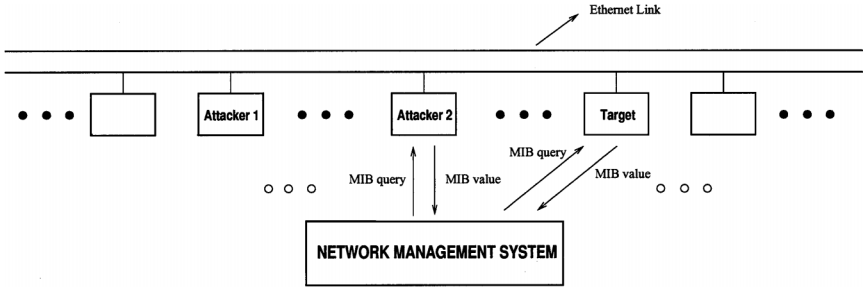


Fig. 4. The research network at NCSU.

Attacker 2 is Sun OS 5.5.1. We used the data corresponding to both attack runs and normal runs as described here.

- *Attack runs*: Three types of DDoS attacks produced by TFN2K (Ping Flood and Targa3) and Trin00 (UDP Flood)—see Criscuolo [16] 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 Fig. 4) 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 3, 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.

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 proactive rules should relate events in T2 or T3 with events in T4 and T5. To illustrate the nature of the MIB variables, Figure 5 depicts the evolution of six relevant MIB variables at the target machine for the Ping Flood attack produced by TFN2K. The variables `icmpInMsgs`, `icmpInEchos`, `icmpInEchoReps` and `icmpOutEchos` are key variables for detecting Ping Flood Attacks, since they are related to the inflow of pings (ICMP Echo Request messages) in the target machine. As shown in Fig. 5, the PING attack is also detectable at the IP layer, through `ipInReceives`. Figure 6 shows the evolution of these same MIBs during the run at the attacker machine, while Fig. 7 shows the evolution of these MIB variables at the target machine, during a normal run. Regarding Fig. 6, it is remarkable that the outbound pings are not detectable through `icmpOutMsgs`, since their addresses were spoofed [16]. Figure 8 depicts `icmpInEchos` at the target, aligned with four MIB variables at the attacker machine that show remarkable activity before

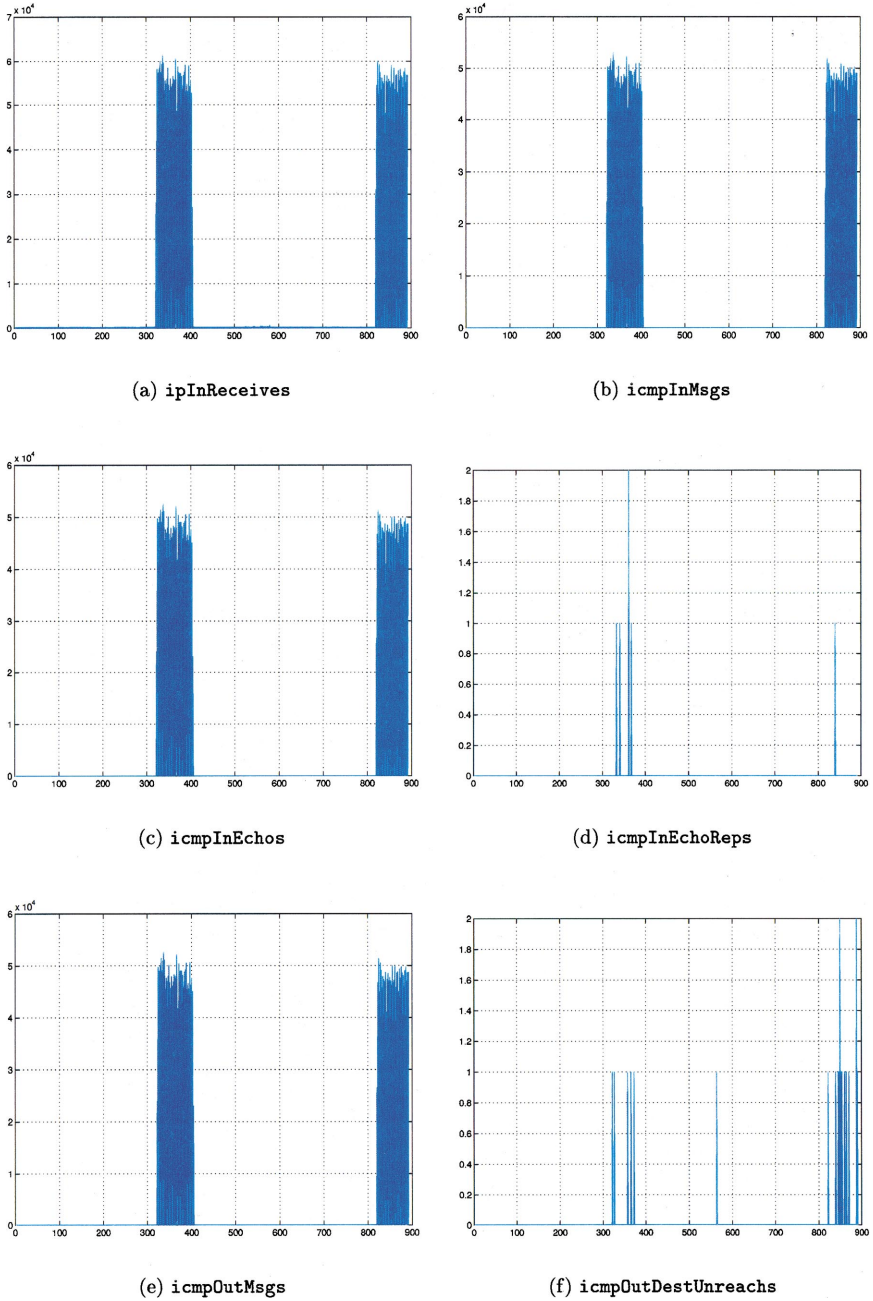
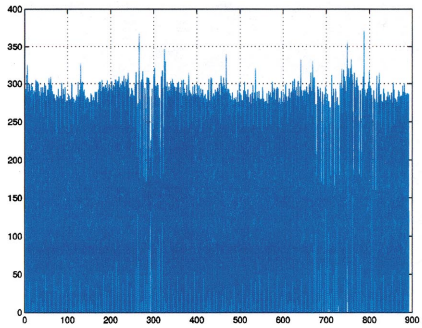
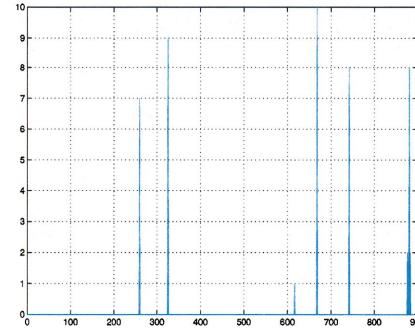


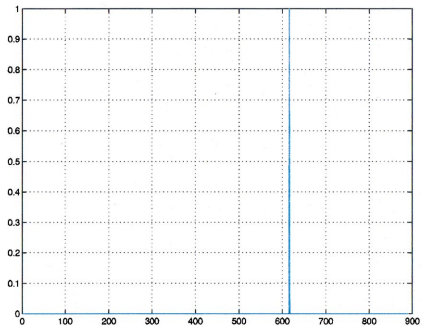
Fig. 5. Evolution of selected MIB variables at the target machine during the TFN2K Ping Flood. The run lasts for 892 samples—74 min. 20 sec. The target is flooded with pings two times during the run.



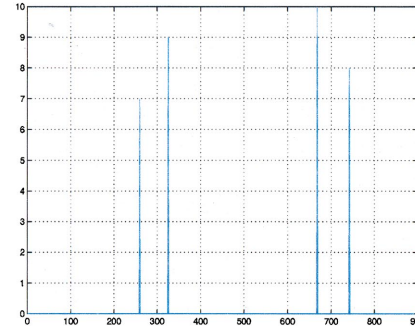
(g) `ipInReceives`



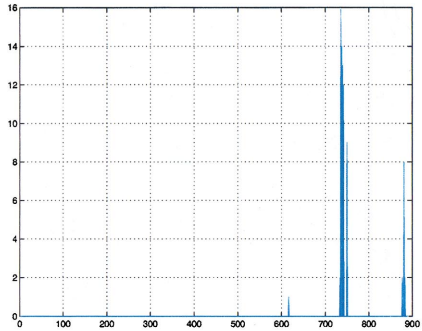
(h) `icmpInMsgs`



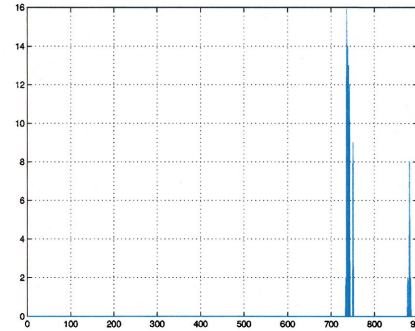
(i) `icmpInEchos`



(j) `icmpInEchoReps`

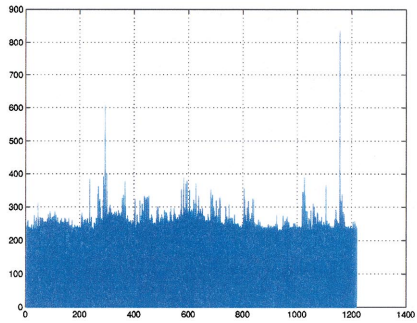


(k) `icmpOutMsgs`

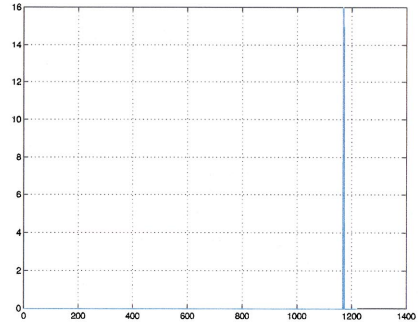


(l) `icmpOutDestUnreachs`

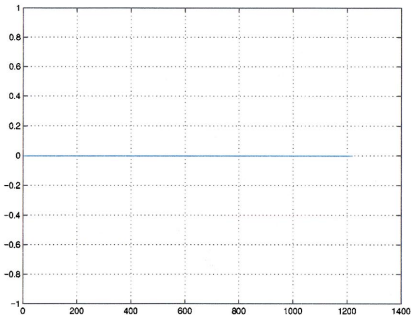
Fig. 6. Evolution of selected MIB variables at the attacker machine during the TFN2K Ping Flood. The run lasts for 892 samples—74 min. 20 sec. `icmpOutMsgs` does not record the outbound pings that constitute the attack, since the pings have spoofed IP addresses.



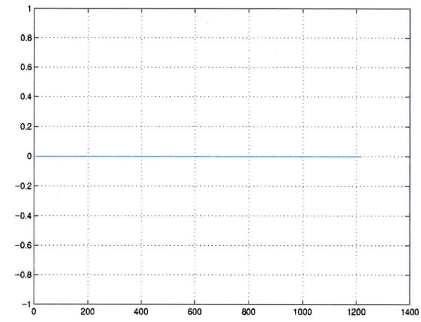
(m) `ipInReceives`



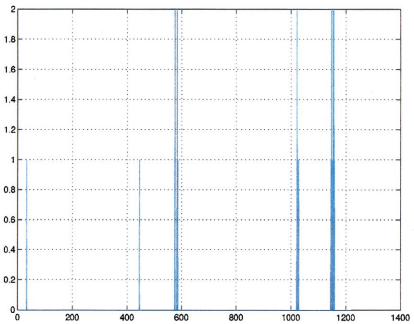
(n) `icmpInMsgs`



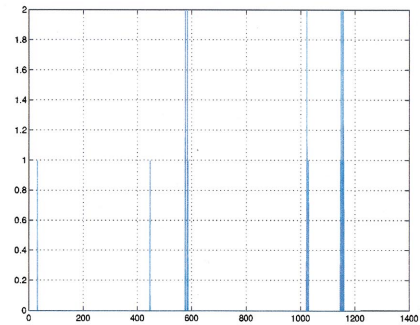
(o) `icmpInEchos`



(p) `icmpInEchoReps`



(q) `icmpOutMsgs`



(r) `icmpOutDestUnreachs`

Fig. 7. Evolution of selected MIB variables at the target machine during a normal run. The run lasts for 1222 samples—101 min. 50 sec. Notice the difference of the scales with respect to Fig. 5.

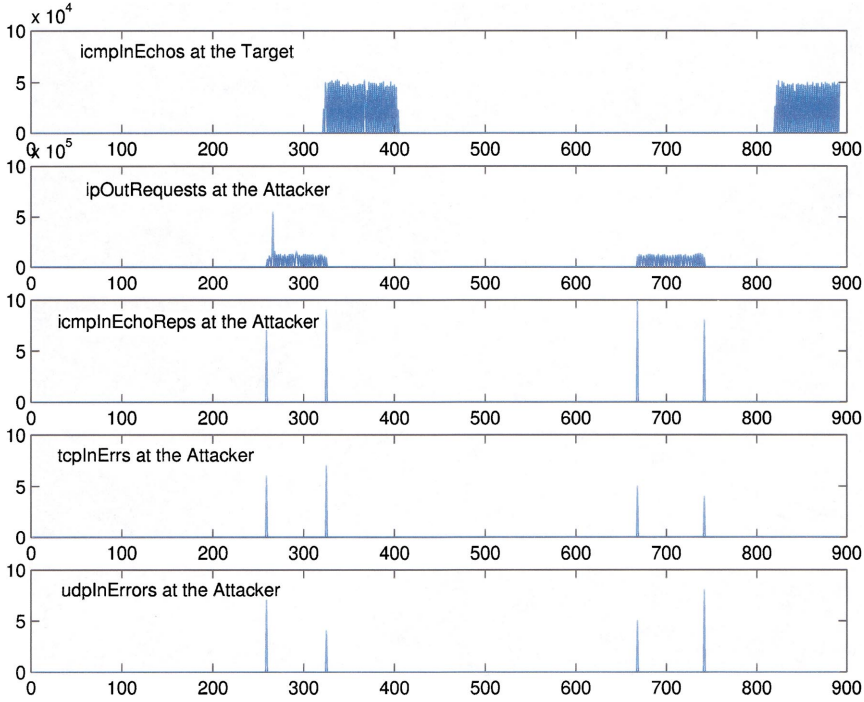


Fig. 8. TFN2K Ping Flood: Selected MIB variables at the attacker and at the target.

the pings reach the target: `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 Fig. 3.
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 Criscuolo [16], 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 *T*2 event, according to Fig. 3.
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 5, and experimental results are presented in Section 6 for the case of the TFN2K Ping Flood Attack, the TFN2K Targa3 attack, and for the Trin00 UDP Flood Attack.

5. EXTRACTING RULES FOR PROACTIVE DETECTION—A METHODOLOGY

5.1. Notation and Definitions

5.1.1. Time Series, Multivariate Time Series, and Collections

A time series is an ordered finite set of numerical values recorded from a variable of interest. It is assumed that the time elapsed between the recording of two consecutive elements is constant. The kth element of the time series $\{z(k)\}$ is denoted as $z(k)$, where $k = 0, 1, \ldots, N-1$ and N denotes the number of elements in $\{z(k)\}$. A multivariate time series of dimension m is an ordered finite set of $m \times 1$ numerical vectors collected from m variables of interest. The kth element of the multivariate time series $\{Z(k)\}$ is denoted as $Z(k) = [z_1(k)\, z_2(k) \cdots z_m(k)]^T$, where $z_i(k)$, $i = 1, 2, \ldots, m$ and $k = 0, 1, \ldots, N-1$ are the kth elements of the individual time series z_i that form Z. It is assumed that all $z_i(k)$ are recorded at the same instant of time, which allows the common index k to be used for their specification. The multivariate time series $\{Z(k)\}$ is represented as an $m \times N$ matrix of numerical variables. We call m the dimension of $\{Z(k)\}$, and N its size. A collection of multivariate time series is a finite set of multivariate time series corresponding to the same variables, but not necessarily having the same size. The jth element of the collection $\mathcal{Z}$ is denoted as Z^j, $j = 1, 2, \ldots, C$. Collections of multivariate time series will be associated with the regimes of operation of the system of interest. In the case of network security, $\mathcal{N}$ corresponds to normal network activity, while $\mathcal{A}$ corresponds to periods of time during which an attack is detected. $C_{\mathcal{N}}$ denotes the number of elements in the collection $\mathcal{N}$, while $C_{\mathcal{A}}$ denotes the number of elements in collection $\mathcal{A}$. Finally, we call the dataset $\mathbb{D}$ as the union of the two collections. $\mathbb{D}$ represents the Training Set, from where knowledge is to be extracted.

5.1.2. Events, Event Sequences, Causal Rules, and Precursor Rules

Events are defined [20] as an ordered pair (A, κ) where $\kappa = 0, 1, 2, \ldots, K-1$ is a time index representing the occurrence time of the event and $A \in \varepsilon$ is an Event Type. ε is a finite set [20], which we call the event alphabet. Event types provide a framework for transforming the raw time series data into more meaningful descriptions. As an example, consider the following procedure for transforming a time series $\{z(k)\}$ having an even size N, into an an event sequence

$\{\epsilon(\kappa)\}$ with size $K = \frac{N}{2}$ and event alphabet $\mathcal{E} = \{E_1, E_2\}$:

- If $[z(k+1) + z(k)] \leq 200$, Then $\epsilon(\kappa) = E_1$, for $\kappa = \frac{k}{2}$ and $k = 0, 2, 4, \ldots, N-2$.
- Otherwise, $\epsilon(\kappa) = E_2$, for $\kappa = \frac{k}{2}$ and $k = 0, 2, 4, \ldots, N-2$.

$z(k)$ can denote the number of alarms issued by a network monitoring device during a single day. A system administrator may only be interested in monitoring a higher level alarm, defined by $\epsilon(\kappa)$, i.e., every couple of days check if more than 200 alarms were issued. If yes, a message is sent. Otherwise, nothing happens. The transformation from the time series space into the event space is the process of time series quantization. The selection of the "right" parameters for performing the quantization depends on the problem at hand. If m time series $\{z_i(k)\}, k = 1, 2, \ldots, m$ are quantized according to the same procedure along the time index, producing m event sequences $\{\epsilon_i(\kappa)\}, i = 1, 2, \ldots, m$, we can define multivariate event sequences and collections of multivariate event sequences the same way we defined their time series counterparts in Section 5.1.1. It is understood that the events $\epsilon_1(\kappa) \in \mathcal{E}_1, \epsilon_2(\kappa) \in \mathcal{E}_2, \cdots \epsilon_m(\kappa) \in \mathcal{E}_m$ are all recorded at the same instant κ, although the individual event alphabets $\mathcal{E}_i, i = 1, 2, \ldots, m$ are not necessarily the same.

Definition 1. (*Causal rule* [13]) If A and B are two events, define $A \overset{\tau}{\Rightarrow} B$ as the rule: If A occurs, then B occurs within time τ. We say that $A \overset{\tau}{\Rightarrow} B$ is a causal rule. □

Definition 2. (*Precursor rule* [12]) If A and B are two events, define $A \overset{\tau}{\Leftarrow} B$ as the rule: If B occurs, then A occured not earlier than τ time units before B. We say that $A \overset{\tau}{\Leftarrow} B$ is a precursor rule. □

Causal rules and precursor rules are special cases of temporal rules, introduced by Agrawal *et al.* [21]. Clearly, the rules $A \overset{\tau}{\Leftarrow} B$ and $A \overset{\tau}{\Rightarrow} B$ are not the same. Notice that B is the antecedent of the precursor rule, while A is the antecedent of the causal rule. Hence, the confidence of the precursor rule—$c(A \overset{\tau}{\Leftarrow} B)$—is the fraction of occurrences of B that were preceded by A within τ units. In the problem at hand, A and B are events recorded at two different event sequences. If $c(A \overset{\tau}{\Leftarrow} B) = 1$, it means that if B occurs, then A always occurred not earlier than τ units before B, and is therefore a precursor of B, in the usual sense of the word. It *does not* mean however that all occurrences of A are followed by B.

The proposed methodology discovers precursor rules of the type $A \overset{\tau}{\Leftarrow} B$ in $\mathbb{D}$, but utilizes the associated causal rule $A \overset{\tau}{\Rightarrow} B$ for detection. The reason for this procedure is clear: we first characterize the security violation (item B) and then search for the precursors (item A). In summary, we mine precursor rules, but apply causal rules.

5.2. Assumptions, Problem Set-Up, Objectives and Procedure

5.2.1. Assumptions

1. The variables are recorded as two collections of multivariable time series of dimension m. Collection $\mathcal{N}$ corresponds to normal operation, while collection $\mathcal{A}$ corresponds to abnormal operation. Typically $C_{\mathcal{A}} \ll C_{\mathcal{N}}$.
2. The m variables can be split into two subsets: Output variables $y_i, i = 1, 2, \ldots, m_1$ and **candidate** input variables $u_i, i = 1, 2, \ldots, m_2$, with $m_1 + m_2 = m$.
3. The output variables are the ones in which the phenomenon of interest manifests itself.
4. The Phenomenon is only observed at collection $\mathcal{A}$.
5. The candidate input variables correspond to variables that may be or may not be related to the occurrence of the phenomenon observed in the output variables.

Given these assumptions, we identify three interrelated problems related to the extraction of knowledge from the dataset $\mathbb{D}$. To simplify our discussion, it is assumed that $m_1 = 1$, i.e., the Phenomenon is only observed on a single output.

Problem 1. *Phenomenon Characterization*: Given the output time series, phenomenon characterization is related to the definition of a suitable event space, through time series quantization. In many problems in computer security the problem of phenomenon characterization is very simple. As shown in Section 4, the output variables related to traffic counting in machines that are targets of Denial of Service attacks records readings of 50,000 units during an attack, compared with about 100 units during normal operation. Also, these bursts are time localized, i.e. the time series in collection $\mathcal{A}$ remain at readings of about 100 units (similar to collections $\mathcal{N}$), except for the bursts characterizing the presence of the attack.

Problem 2. *Identifying the Input Variables*: The objective is to select which among the m_2 variables contain precursors for the phenomenon observed in the output. The objective is to obtain time series for which high confidence precursor rules of the type $A \overset{\tau}{\Leftarrow} B$ exist, where B is the phenomenon characterized in Problem 1, while A is an event extracted from the candidate time series. Notice that τ is not known. Hence, it is not advisable at this stage to do time series quantization at the candidate inputs, as valuable precursor information may be destroyed in the process. Clearly, we need a procedure capable of performing the following two tasks: **(1) Detection:** Given an input-output pair $(\{u(k)\}, \{y(k)\})$ measure the likelihood that a rule of the type $A \overset{\tau}{\Leftarrow} B$ exists, where A is a precursor extracted from $\{u(k)\}$ and B is a phenomenon extracted from $\{y(k)\}$ *without knowing the true nature of the precursor,* or *the delay between precursor and phenomenon*; **(2) Gradation:** Given a fixed output and m_2 candidate inputs, rank candidate input

variables according to the likelihood that precursor rules exist, *without knowing the true nature of the precursor, neither the delay between precursor and phenomenon.* We show in the sequel that the Granger Causality Test is an adequate procedure for addressing both tasks.

Problem 3. *Precursor Characterization*: Given the input variables that are most likely to contain precursors, the problem of precursor characterization is to extract the precursors as time-localized occurrences in the time series through a process of time series discretization of the same nature as Problem 1. The key point however, is that following the solution of Problem 2, one has evidence that these time-localized occurrences give rise to event types that are related to the phenomenon at the output through a rule of the type $A \overset{\tau}{\Leftarrow} B$.

5.2.2. Procedure

Based on the above, we suggest the following procedure for extracting precursor rules relating phenomenon in the outputs with precursors at candidate inputs:

- **Step 1:** Solve Problem 1 through adequate time series quantization at the output time series.
- **Step 2:** Solve Problem 2 by applying the Granger Causality Test (GCT) to all input-output pairs $\{(u_i(k),\ y(k))\}$, $i = 1, 2, \ldots, m_2$, and compute the GCI g_i corresponding to each candidate input. Select candidate inputs either by setting a threshold on g_i, or by choosing the top v scores, where typically $v \ll m_2$.
- **Step 3:** Solve Problem 3 through adequate time series quantization of the input time series selected on Step 2. The objective is to extract time-localized structures at the selected inputs that precede the phenomenon in the output. Discard input time series that do not show time-localized structure preceding the phenomenon. At the end of this step, one has determined the **Phenomenon** and **Precursor** events of interest, as well as a number of candidate precursor rules of the form **Precursor** $\overset{\tau}{\Leftarrow}$ **Phenomenon**.
- **Step 4:** Compute the confidence of the *associated* causal rules **Precursor** $\overset{\tau}{\Rightarrow}$ **Phenomenon**, and select the best ones either by thresholding or ranking. At this step we are verifying if indeed the Precursor events at the inputs are preceding the phenomenon events at the output.

In the next section we will describe the Granger Causality Test, and discuss its suitability as an exploring tool for knowledge discovery, targeted on Step 2. A more complete discussion of the theoretical developments can be found in Cabrera and Mehra [12]. In Section 6, we describe the results obtained when applying this procedure for the extraction of precursor rules relating attacking nodes with target nodes in various types of Distributed Denial of Service attacks.

5.3. The Granger Causality Test as an Exploratory Tool

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 the variable y. If not, then u does not Granger-cause y. The Granger Causality Test (GCT [22]) 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 a_i and b_i parameters ($1 \leq i \leq p$) in the following unrestricted equation:

$$y(k) = \sum_{i=1}^{p} \alpha_i y(k-i) + \sum_{i=1}^{p} \beta_i u(k-i) + e_1(k) \tag{5.1}$$

Parameter estimation is performed using Ordinary Least Squares (OLS) [23]. If $\{y(k)\}$ and $\{u(k)\}$ are time series of size N, it results on a regression with $T := N - p$ equations, out of which $2p$ parameters are estimated. The computational cost of the procedure is $O(T^2)$. The Null Hypothesis H_0 of the GCT is given by:

$$H_0: \quad \beta_i = 0, \qquad i = 1, 2, \ldots, p$$

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

$$y(k) = \sum_{i=1}^{p} \delta_i y(k-i) + e_0(k) \tag{5.2}$$

Again, estimation of the δ parameters lead to an OLS problem with T equations. The procedure of the GCT is as follows. Let R_1 and R_2 denote the sum of the squared residuals under the two cases:

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

If the Granger Causality Index (GCI) g given by:

$$g = \frac{(R_0 - R_1)/p}{R_1/(T-2p-1)} \sim F(p, T-2p-1) \tag{5.3}$$

is greater than the specified critical value for the F-test, then reject the null hypothesis that u does not Granger-cause y. Here, $F(v_1, v_2)$ denotes an F distribution with parameters v_1 and v_2 [24]. As g increases, the p-value associated with the pair $(\{u(k)\}, \{y(k)\})$ decreases, lending more evidence that the Null Hypothesis is *false*. In other words, high values of g are to be understood as representing strong evidence that u is causally related to y. (Note: The p-value of a Statistical Test is the smallest significance level that leads to the rejection of the Null Hypothesis [25]). In this work, we utilize the GCT in a exploratory manner, to

compare the causality strength of two candidate input time series with respect to a given output. Following the p-value interpretation, we say that $\{u_1(k)\}$ is more likely to $\{u_2(k)\}$ to be causally related with $\{y(k)\}$ if $g_1 > g_2$, where $g_i, i = 1, 2$ denote the GCI for the input-output pair (u_i, y). We may be interested in selecting the top 5 or 10 individual candidate input time series that are more likely to be causally related to $\{y(k)\}$ for more detailed inspection. It is shown [12] that under certain idealized conditions and suitable approximations, the GCI corresponding to the input-output pair (u, y) is proportional to the confidence of the **Precursor** $\overset{\tau}{\Leftarrow}$ **Phenomenon** rules to be extracted from this input-output pair.

6. PROACTIVE DETECTION OF DDoS ATTACKS: EXPERIMENTAL RESULTS

6.1. Step 1: Phenomenon Characterization—Detecting Attacks

1. **TFN2K Ping Flood** The Ping Flood attack is effected by sending a large amount of ICM-PECHOREQUEST packets to the target. Clearly, `icmpInEchos` is the output variable.
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 can be used as the output. 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 output in this case.

Step 1 corresponds to constructing a passive intrusion detection model. While we used domain knowledge in the present study, we examined the problem of designing passive intrusion detectors using MIB variables [9,10]. Data mining, information theory, and clustering techniques were used to develop the model. The results of the experiments revealed that MIB variables can be successfully used for detecting traffic-flood Denial of Service Attacks, of the type examined in this paper.

Remark 1. (*Time Series Segmentation and Fault Detection in Networks*) Time series segementation was employed for detecting anomalies in network operation, due to component faults [26,27]. In both references, anomalies were detected as variations on the parameters of AutoRegressive Models.

6.2. Step 2: Extracting Inputs Containing Precursors

We apply the GCT, by comparing the residuals of the AR Model corresponding to the outputs—Key Variables at the Target, with the ARMA Models

Table I. Key Variables at the Attacker for TFN2K—Ground Truth

MIB	Event
`icmpInEchoReps`	*T*2
`tcpInErrs`	*T*2
`tcpInSegs`	*T*2
`udpInErrors`	*T*2
`udpInDatagrams`	*T*2
`ipOutRequests`	*T*3

corresponding to the input-output pairs where the outputs are the Key Variables 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 Criscuolo [16], we have domain knowledge (ground truth) about the Key Variables at the Attacker, i.e., the variables that contain precursors for the attacks. Tables I and II display these variables. We have applied the GCT for the two runs of each of the three types of DDoS attacks. *T*4 events happen more than once in each run; for example, as shown in Fig. 8, 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 III gives the relevant information for each of the runs, the parameters for the GCT, and the thresholds for each case.

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 Tables I and II are correctly picked by the GCT. False alarms correspond to MIB variables being flagged in the decoys. Tables IV–XV display the overall results, which are summarized in Table XVI. As an example, Notice that in Run 1 for the TFN2K Ping Flood

Table II. Key Variables at the Attacker for Trin00—Ground Truth

MIB	Event
`udpInDatagrams`	*T*2
`udpOutDatagrams`	*T*3
`ipOutRequests`	*T*3

Table III. Statistics for the Attack Runs, Parameters for the GCT, and Thresholds for the g Statistic for Two Significance Levels[a]

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

[a]These thresholds are the critical levels corresponding to $F(p, T - 2p - 1)$.

Table IV. TFN2K Ping Flood Run 1: Top MIBs at the Attacker According to the g Statistic

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 V. TFN2K Ping Flood Run 1: Performance of the GCT for Normal Runs at the Attacker[a]

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

[a]The significance level is 95%.

Table VI. TFN2K Ping Flood Run 2: Top MIBs at the Attacker According to the g Statistic

Rank	MIB	g
1	`icmpInMsgs`	1.45
2	`icmInEchoReps`	1.45

Table VII. TFN2K Ping Flood Run 2: Performance of the GCT for Normal Runs at the Attacker[a]

Run	Max g	No. sign. MIBs
1	5.86	6
2	1.15	0
3	—	—
4	1.11	0
5	1.40	1
6	0.88	0
7	—	—
8	5.34	7

[a]The significance level is 95%.

Table VIII. TFN2K Targa 3 Run 1: Top MIBs at the Attacker According to the g Statistic

Rank	MIB	g
1	`ipInDelivers`	1.70
2	`udpOutDatagrams`	1.70
3	`udpInDatagrams`	1.70
4	`ipInReceives`	1.68
5	`udpInErrors`	1.51
6	`ipOutRequests`	1.32

Table IX. TFN2K Targa 3 Run 1: Performance of the GCT for normal runs at the Attacker[a]

Run	Max g	No. sign. MIBs
1	0.96	0
2	5.61	3
3	1.86	5
4	1.17	0
5	1.14	0
6	1.09	0
7	3.55	11
8	1.22	0

[a]The significance level is 95%.

Table X. TFN2K Targa3 Run 2: Top MIBs at the Attacker According to the g Statistic

Rank	MIB	g
1	`udpInErrors`	1.28

Table XI. TFN2K Ping Targa3 Run 2: Performance of the GCT for normal runs at the Attacker[a]

Run	Max g	No. Sign. MIBs
1	0.89	0
2	4.92	3
3	3.60	4
4	1.19	0
5	1.15	0
6	1.25	0
7	—	—
8	1.55	4

[a]The significance level is 95%.

Table XII. Trin00 UDP Flood Run 1: Top MIBs at the Attacker According to the g Statistic

Rank	MIB	g
1	`icmpInMsgs`	3.57
2	`icmpInDestUnreachs`	3.56
3	`udpOutDatagrams`	2.70
4	`ipOutRequests`	2.70
5	`icmpOutMsgs`	2.64
6	`icmpOutDestUnreachs`	2.62
7	`tcpRetransSegs`	1.98
8	`udpInDatagrams`	1.81
9	`ipInDelivers`	1.79
10	`ipInReceives`	1.72
11	`tcpCurrEstab`	1.56

Table XIII. Trin00 UDP Flood Run 1: Performance of the GCT for Normal Runs at the Attacker[a]

Run	Max g	No. sign. MIBs
1	1.23	0
2	1.70	3
3	2.10	5
4	1.76	3
5	1.33	1
6	1.35	2
7	7.04	10
8	3.69	2

[a]The significance level is 95%.

Table XIV. Trin00 UDP Flood Run 2: Top MIBs at the Attacker According to the g Statistic

Rank	MIB	g
1	`icmpOutMsgs`	7.96
2	`icmpOutDestUnreachs`	7.94
3	`icmpInMsgs`	3.73
4	`icmpInDestUnreachs`	3.73
5	`tcpPassiveOpens`	3.04
6	`ipOutRequests`	2.94
7	`udpOutDatagrams`	2.94
8	`tcpCurrEstab`	2.78
9	`tcpInSegs`	2.38
10	`tcpRetransSegs`	2.30
11	`tcpOutSegs`	2.27
12	`ipInDelivers`	2.24
13	`udpInDatagrams`	2.24
14	`udpNoPorts`	2.13
15	`ipInReceives`	2.12
16	`tcpActiveOpens`	1.96

Table XV. Trin00 UDP Flood Run 2: Performance of the GCT for Normal Runs at the Attacker[a]

Run	Max g	No. sign. MIBs
1	3.67	2
2	4.10	5
3	3.76	8
4	1.20	0
5	1.54	2
6	4.09	9
7	—	—
8	2.77	8

[a]The significance level is 95%.

Table XVI. Results of Step 2: Detection Rates and FA Rates for MIB Variables that Contain Precursors to DDoS Attacks

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

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 (e.g., [19]) 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 XVI, 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 V indicates 23 out of 64×8, giving the FA rate of 4.49% recorded in Table XVI.

6.3. Steps 3 and 4: Precursor Characterization and Determining the Confidence of the Causal Rules

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 the precursors. As discussed in Section 5, we have a problem of time series quantization. We looked 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 at the *Normal Profile of Jumps*. Key Attacker Variables with no Key Events are discarded. As shown in Table XVII, 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. Concerning Step 4, 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.

Table XVII. Final Results: Detection Rates and FA Rates for Events at MIB Variables

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

7. APPLICATIONS ON SECURITY MANAGEMENT

As noted by Subramanian [8], security management is both a technical and an administrative consideration in information management. In this section we describe three possible areas where Proactive Intrusion Detection can be applied in security management.

Active Response The most obvious utilization of the technology is to trigger an early, or pre-emptive response to a security violation [28]. To fix ideas, let us consider a scenario involving three network clusters, which will be monitored by three individual NMM (Network Management Modules), installed on machines A1, A2, and A3 on each cluster. Assume that DDoS slaves are installed in A1 and A2, and the target is somewhere in Cluster 3. Assume also that Proactive Rules were obtained, relating precursors at A1 and A2 with an attack against Cluster 3. Assume finally that the master is somewhere in the Internet, not monitored by the NMMs in the clusters. Following the detection of precursors in A1 and A2, we identify three general classes of responses:

- **Class 1: Local Active Response at the Destination.** The NMM in Cluster 1 and Cluster 2—NMM1 and NMM2—inform NMM3 that an attack against Cluster 3 is imminent. It is assumed that a Virtual Private Network (VPN) is in place, that carries messages in a faster and more reliable way than the "raw" Internet used to carry the attack. NM3 takes a series of actions *inside* Cluster 3, i.e., sending instructions to firewalls, disabling router ports, etc. NMM3 does not try to interact with the other clusters—this is done in Class 3—but simply protect itself against the incoming attack.
- **Class 2: Local Active Response at the Source.** NMM1 and NMM2 thwart the attack, by shutting down A1 and A2. NMM3 is not even made aware that an attack from clusters 1 and 2 was under way. This may be desirable in many cases, taking into account the legal aspects involved in Information Warfare. Recall that DDoS Slaves are cloaked inside the hosts; their presence is first detected by the attack precursors. Recall that the precursors discovered by the method described in Section 5 are intrinsic to the attack; the hacker could not avoid leaving those traces. The hacker may not be even

aware that NMS can capture the precursors. Recall also that the precursors were found *automatically* by our method, out of a relatively large universe of possibilities.

- **Class 3: Global Active Response.** NMM1 and NMM2 inform NMM3 about the imminent attack against cluster 3. NMM3, relying on a properly designed messaging system and a VPN, severs the connection between cluster 1, cluster 2, and the rest of the world. Alternatively, the decision of shutting down the exits from cluster 1 and cluster 2, might be made in agreement with NMM1 and NMM2. One can wonder why NMM1 and/or NMM2 would bring such an outcome against themselves. The reason is that the subnetwork formed by NMM1, NMM2 and NMM3 is in fact a neutral agent in the whole process. NMMi is not in league with Cluster i, but part of an overall defensive system, aimed to protect the network as a whole (cluster 1, cluster 2, and cluster 3). The practicality and technical hurdles involved in setting such an infrastructure is, of course, a subject of research. Coordination among different agents for responding to a DDoS attack has been investigated [29].

Computer Forensics. Computer Forensics, is an emerging discipline lying at the intersection of Information Assurance and Law Enforcement (e.g., [5]). Computer Forensics starts with the fact of abuse having occurred, and attempts to gather the evidence needed by the investigators to identify the culprits, and allow their prosecution. The detection rules and interpretation rules extracted during the Training Stage, can be used in Forensic Computing. These rules can be understood as "statistical signatures" of the security violations; if these signatures are encountered on-line, they will trigger the detection rules. If they are encountered off-line, they can be used as evidence that the security violation took place.

Pruning False Alarms from Passive Intrusion Detectors. Current passive IDSs are plagued by high rates of false alarm [3,30]; explainable in part by the base rate fallacy of classical statistics [31], a result of the rarity of attacks in comparison to normal activity. The presence (or, rather, the absence) of reliable precursors may be used to prune false alarms from passive IDSs. Intuitively, an alarm raised by a passive IDS which is preceded by an identified precursor should be given more weight than an alarm that is not preceded by a precursor. This is a typical situation where the NMS can be seen as a repository of valuable information about the environment surrounding an IDS. Current IDS research is moving towards the development of cost-sensitive schemes, balancing accuracy, resource utilization, and the damage cost of intrusions [32,33]. On a probabilistic sense [32], the precursor enables the NMS to update the IDS regarding the prior probability of a given Security Violation. Clearly, extensive experimentation is needed to validate the concept.

8. CONCLUSIONS

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. We are currently experimenting with a larger networked testbed, including six hosts and three routers, instrumented by a Network Management System. Our results will appear in the near future.

Finally, to be applicable in a realistic setting, multiple domains are likely to be involved, as suggested in Section 7. Thus, our plans for further work are to scale the approach to multi-domain environments. Two issues need to be investigated: (i) network management systems in multiple domains will have to be in communication (e.g., [34,35]): 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.

ACKNOWLEDGMENTS

This work was supported by the Air Force Research Laboratory (Rome, New York) under contracts F30602-00-C-0126 and F30602-01-C-057 to Scientific Systems Company, and by Aprisma's University Fellowship Program 1999/2000. Scientific Systems Company acknowledges the continuing support from the Defensive Information Warfare Branch at the Air Force Research Laboratory in Rome, New York. We are particularly grateful to Mr. Peter J. Radesi and Dr. Leonard J. Popyack, Jr. from AFRL, for their encouragement.

REFERENCES

1. D. Denning, An intrusion detection model, *IEEE Transactions on Software Engineering*, Vol. 13, No. 2, pp. 222–232, February 1987.
2. B. Schneier, *Secrets and Lies: Digital Security in a Networked World*, John Wiley, 2000.
3. S. Kent, On the trail of intrusions into information systems, *IEEE Spectrum*, pp. 52–56, December 2000.
4. F. B. Schneider, ed., *Trust in Cyberspace*, National Academy Press, 1998.
5. J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel, and E. Stoner, State of the practice of intrusion detection technologies, Technical Report CMU/SEI-99-TR-028, Carnegie Mellon University, Software Engineering Institute, January 2000.

6. L. Lewis, *Managing Business and Service Networks*, Kluwer Academic Press, 2001.
7. E. Amoroso, *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Traps, Trace Back and Response*, Intrusion. Net Books, First Edition, 1999.
8. M. Subramanian, *Network Management—Principles and Practice*, Addison-Wesley, 2000.
9. X. Qin, W. Lee, L. Lewis, and J. B. D. Cabrera, Integrating intrusion detection and network management, *Proceedings of the Eighth IEEE/IFIP Network Operations and Management Symposium*, Florence, Italy, pp. 329–344, April 2002.
10. X. Qin, W. Lee, L. Lewis, and J. B. D. Cabrera, Using MIB II variables for network intrusion detection. D. Barbará and S. Jajodia, eds., *Applications of Data Mining in Computer Security*, Kluwer Academic Publishers, Boston, 2002 (in press).
11. J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, and R. K. Mehra, Proactive intrusion detection—A study on temporal data mining. D. Barbará and S. Jajodia, eds., *Applications of Data Mining in Computer Security*, Kluwer Academic Publishers, Boston, 2002 (in press).
12. J. B. D. Cabrera and R. K. Mehra, Extracting precursor rules from time series—A classical statistical viewpoint, *Proceedings of the Second SIAM International Conference on Data Mining*, Arlington, Virginia, pp. 213–228, April 2002.
13. G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, Rule discovery from time series, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 16–22, 1998.
14. Science and Technology Section, Internet security—Anatomy of an attack, *The Economist*, pp. 80–81, February 19, 2000.
15. D. Moore, G. M. Voelker, and S. Savage, Inferring Internet Denial-of-Service Activity, *Proceedings of USENIX Security Symposium*, Washington, D.C., 2001.
16. P. J. Criscuolo, Distributed denial of service—Trin00, Tribe flood network, tribe flood network 2000, and Stacheldraht, Technical Report CIAC-2319, Department of Energy—Computer Incident Advisory Capability, (CIAC) February 2000.
17. K. Kendall, A database of computer attacks for the evaluation of intrusion detection systems, Master's thesis, Massachusetts Institute of Technology, June 1999.
18. J. B. D. Cabrera, B. Ravichandran, and R. K. Mehra, Statistical traffic modeling for network intrusion detection, *Proceedings of the Eighth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, IEEE Computer Society, San Francisco, California, pp. 466–473, August 2000.
19. W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, pp. 363–388, 1994.
20. H. Mannila, H. Toivonen, and A. I. Verkamo, Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, Vol. 1, No. 3, pp. 259–289, 1997.
21. R. Agrawal, T. Imielinski, and A. Swami, Database mining: A performance perspective, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, pp. 914–925, December 1993.
22. C. W. J. Granger, Investigating causal relations by econometric models and cross-spectral methods, *Econometrica*, Vol. 34, pp. 424–438, 1969.
23. J. Hamilton, *Time Series Analysis*, Princeton University Press, 1994.
24. M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions.* John Wiley, New York, Second Edition, 1993.
25. G. Casella and R. L. Berger, *Statistical Inference*, Duxbury Press, Belmont, California, p. 364, 1990.
26. M. Thottan and C. Ji, Proactive anomaly detection using distributed agents, *IEEE Network*, pp. 21–27, September 1998.
27. F. Zhang and J. Hellerstein, An approach to on-line predictive detection, *Proceedings of the Eighth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, IEEE Computer Society, San Francisco, California, pp. 549–556, August 2000.

28. J. B. D. Cabrera, L. J. Popyack, Jr., L. Lewis, B. Ravichandran, and R. K. Mehra, The monitoring, detection, interpretation and response paradigm for the security of battlespace networks, *Proceedings of IEEE MILCOM 2001*, Washington, D.C., October 2001.
29. D. Schnackenberg, K. Djahandari, and D. Sterne, Infrastructure for intrusion detection and response, *Proceedings of DARPA Information Survivability Conference and Exposition*, Hilton Head Island, South Carolina, January 2000.
30. S. Northcutt, *Network Intrusion Detection—An Analyst's Handbook*, New Riders Publishing, 1999.
31. S. Axelsson, The base-rate fallacy and its implications for the difficulty of intrusion detection, *Proceedings of the Sixth ACM Conference on Computer and Communications Security*, Singapore, November 1999.
32. J. E. Gaffney, Jr., and J. W. Ulvila, Evaluation of intrusion detectors: A decision theory approach, *Proceedings of the IEEE Symposium on Security and Privacy*, May 2001.
33. W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok, Toward cost-sensitive modeling for intrusion detection and response, *Journal of Computer Security*, 2002 (in press).
34. K. Boudaoud, H. Labiod, R. Boutaba, and Z. Guessoum, Network security management with intelligent agents, IEEE Publishing, *Proceedings of NOMS*, 2000.
35. 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, IEEE Publishing, *Proceedings of NOMS*, 2000.

João B. D. Cabrera was born in Rio de Janeiro, Brazil. He received the Ph.D. degree in electrical engineering from Yale University, USA, the M.Eng. degree in control engineering from the Tokyo Institute of Technology, Japan, and the B.S. degree in electrical engineering from ITA—the Aeronautical Institute of Technology, Brazil. Since December 1996 he has been with Scientific Systems where he is currently a Senior Research Engineer, involved in projects related to information assurance, statistical pattern recognition and machine learning.

Lundy Lewis is the Director of Research at Aprisma Management Technologies in the USA. He holds sixteen patents in the area of network and service management, and has written three books in the area: *Managing Computer Networks: A Case-Based Reasoning Approach* (Artech House, 1995), *Service Level Management for Enterprise Networks* (Artech House, 1999), and *Managing Business and Service Networks* (Kluwer/Plenum 2001). He is a frequent speaker at professional shows and conferences. He holds a Ph.D. in Philosophy from the University of Georgia, an M.S. in Computer Science from Rensselaer Polytechnic Institute, and a B.S. in Mathematics and B.A. in Philosophy from the University of South Carolina. He is an adjunct professor in the Computer Science department at the University of New Hampshire.

Xinzhou Qin received the B.S. degree in Electrical Engineering from Beijing Polytechnic University, China, in 1996. He received the M.S. degree in Electrical Engineering from North Carolina State University, in 1998. He is currently a Ph.D. student in computer science at College of Computing at Georgia Institute of Technology, USA. His research interests are network and system security and network management.

Wenke Lee is an Assistant Professor in the College of Computing at Georgia Institute of Technology. He received his Ph.D. in Computer Science from Columbia University and his B.S. in Computer Science from Zhongshan University, China. His research interests include network security, data mining, and network management. He received a Best Paper Award (in applied research category) at the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-99) in 1999, and a NSF CAREER Award in 2002.

Raman K. Mehra is the founder, and has been the CEO and President of Scientific Systems since 1976. He received his Ph.D. from Harvard University in 1968 in Engineering and Applied Mathematics and a B.E. in Electrical Engineering from Punjab University (India) in 1964. He was a Gordon McKay Associate Professor of Engineering and Applied Mathematics at Harvard University from 1972 to 1976. He has published over 250 papers in his field, and edited a book on System Identification. He was given the Eckman Award in 1971 by the American Automatic Control Council for research contributions under the age of 30 and was elected as a Fellow of the IEEE in 1986 for contributions to the development of theories of identification, estimation and optimal control and their applications in aerospace and industrial systems.