

# Hi-DRA: Intrusion Detection for Internet Security

Richard A. Kemmerer, *Fellow, IEEE*, Giovanni Vigna, *Member, IEEE*

**Abstract**—Intrusion detection systems monitor computer networks looking for evidence of malicious actions. Networks are complex systems and a comprehensive intrusion detection solution has to be able to manage event streams with different content, speed, level of abstraction, and accessibility. Therefore, it is necessary to distribute intrusion detection sensors across multiple protected networks, manage their configuration as the security posture of the networks changes, and process the results of their analysis so that a high-level picture of the security state of the network can be provided to the administrators. This paper presents Hi-DRA, a network surveillance, analysis, and response system for high-speed, wide-area networks. The system provides a framework for the modular development of intrusion detection sensors in heterogeneous, high-speed environments. In addition, the system provides an infrastructure that supports the dynamic configuration of the sensors and the collection and interpretation of their results. The system, as a whole, is able to provide fine-grained monitoring across wide-area networks and, at the same time, is able to correlate the results of the analysis of the different sensors into a high-level expressive description of security violations.

**Index Terms**—Intrusion Detection, Misuse Detection, Anomaly Detection, Alert Correlation, Security, Computer Security, Network Security

## I. INTRODUCTION

In recent years networks have become larger, faster, and highly dynamic. In particular, the Internet, the world-wide TCP/IP network, has become a mission-critical infrastructure for governments, companies, financial institutions, and millions of everyday users.

Large-scale infrastructures that provide nation-wide mission-critical services are usually managed (or at least regulated) by a national central authority. An example is the US national power grid. In contrast, the Internet by its construction is a *network of networks* composed of autonomous subsystems managed by companies, organizations, and governments with different and sometimes conflicting goals. Thus, mission-critical networks must survive and interoperate within an untrusted heterogeneous environment.

This model suggests that the protection of the network infrastructure must rely on *local surveillance* and *global coordination and control*. Local surveillance addresses the problem of securing a protection domain by means of proactive security measures, extensive monitoring, and real-time response. But local protection is not enough. Worms and distributed denial-of-service attacks have shown that the future threats to the infrastructure will involve numerous protection domains as victims or unwilling collaborators. Therefore, there is a need

to create a nation-wide security infrastructure that enables the correlation of security-related information coming from different subsystems to obtain a global view of the security state of the infrastructure and that enables command and control capabilities from a central or distributed control station.

The ability to analyze and reconfigure the security posture of a network from a single control component is particularly important when the infrastructure is under attack “as a whole,” for example in the case of cyber-terrorism. In these cases, the ability to integrate the information coming from different parts of the network, to coordinate countermeasures, and possibly to counterattack is vital. Unfortunately, existing approaches to network monitoring and surveillance suffer from a number of limitations.

- Current techniques for network monitoring and intrusion detection are not able to cope with the increasing speed of networks and are mainly focused on end-to-end attacks.
- Monitors and surveillance tools cannot be dynamically configured to respond in real-time to new threats and to changes in the level of concern.
- Monitoring is performed without taking into account the characteristics of the protected network such as topology and deployed services.
- There is no large-scale coordination and control infrastructure that allows one to correlate surveillance security reports and exert control over the type and level of surveillance to be performed in the protected networks.

At UCSB we have developed a network surveillance, analysis, and response system for high-speed, wide-area networks that will overcome these limitations. The system, which is called Hi-DRA (*High-speed, wide-area network Detection, Response, and Analysis*), provides enhanced surveillance, analysis, and response capabilities in the context of high-speed, wide-area networks. The Hi-DRA architecture is presented schematically in Figure 1. The figure represents three protected networks instrumented with Hi-DRA surveillance components and connected by the Hi-DRA wide-area communication and control infrastructure.

The protected networks are named *digi.com*, *univ.edu*, and *devel.gov*. The network monitoring surveillance tool is represented in detail for network *devel.gov*, which is connected to the Internet (the oval at the center of the figure) through a high-speed link (the thick black line in the figure). Traffic on the link is partitioned into sub-portions and assigned to dedicated links. On each link a set of network sensors performs network analysis. The process of partitioning the network traffic and assigning analysis tasks to network sensors is described in Section II. In addition to sensors that monitor the traffic on the network up-link, the *devel.gov* network is instrumented with both host-based and network-based sensors. These sensors are configurable and controllable surveillance

Manuscript received March 25, 2005; revised April 6, 2005.

R.A. Kemmerer and G. Vigna are with the Reliable Software Group at the University of California, Santa Barbara, CA 93106 USA (email [kemmer,vigna]@cs.ucsb.edu).

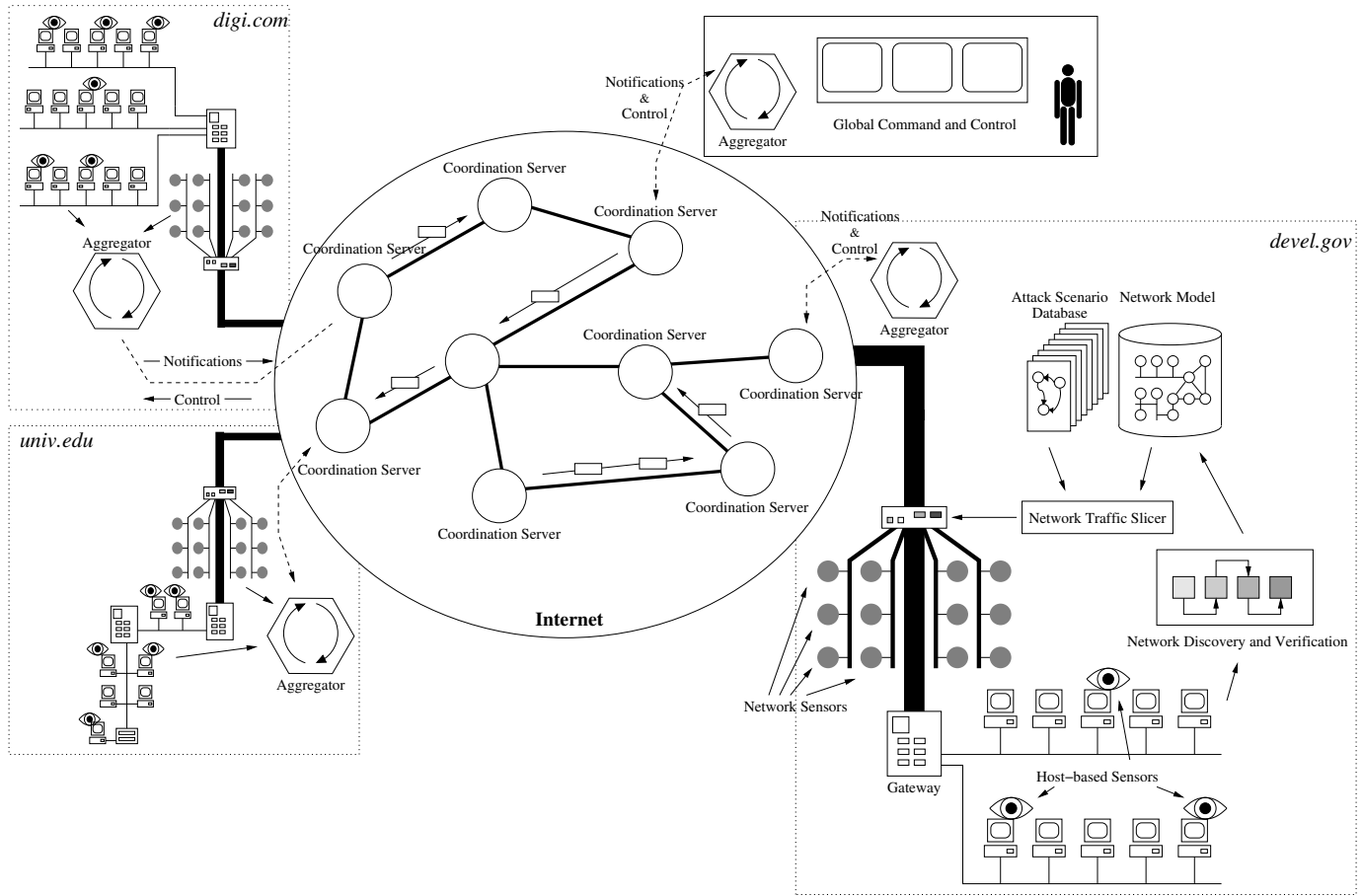


Fig. 1. Hi-DRA high-level architecture.

monitors, whose characteristics are detailed in Section III. Both the configuration of the sensors and the traffic partitioning algorithm are closely tailored to the protected network. This is made possible by the *network model* component. This component contains security-critical information about the protected network. The model structure and the network discovery and validation tools are described in [1]. An instance of the model is completed by means of a set of composable network tools. The information gathered is used as a basis for configuration and customization of the system. In addition, the same tools can be used to validate the information stored in the model against the actual configuration of the network. This process allows the security administrator to identify misconfigurations and possible vulnerabilities in the protected network. Finally, the surveillance monitors that are local to each protected network are coordinated through Hi-DRA's wide-area communication and control infrastructure, which is based on the Siena system [2]. Each network is connected to the infrastructure by means of an *aggregator* component (represented as a hexagon in the figure). Messaging is then managed by a number of interconnected coordination servers deployed across the Internet (the circles inside the Internet oval) that use Siena to implement a scalable and survivable communication infrastructure. The infrastructure, is also used by one or more Global Command and Control

monitors, which receive alerts from the protected networks, perform high-level situation analysis and course of action determination, and send control messages to the protected networks using the infrastructure.

Due to space limitations we can only highlight a few of the components of Hi-DRA. Therefore, we will limit our presentation to the partitioning technique that we use to deal with high-speed networks, our “web of sensors” approach to providing highly configurable sensors, and our approach to intrusion detection alert correlation.

The remainder of this paper is structured as follows: Section II presents an overview of our partitioning approach to dealing with high-speed networks. Section III discusses our “web of sensors” approach to managing and controlling highly configurable network and host-based sensors. Section IV provides an overview of the components of the correlation process and discusses the tool that implements this process. Section V presents related work. Finally, Section VI draws conclusions and outlines future work.

## II. MONITORING HIGH SPEED LINKS

Network-based intrusion detection systems (NIDSs) perform security analysis on packets obtained by eavesdropping on a network link. The constant increase in network speed and throughput poses new challenges to these systems. Current

network-based IDSs are barely capable of real-time traffic analysis on saturated Fast Ethernet links (100 Mbps) [3]. As network technology presses forward, Gigabit Ethernet (1000 Mbps) has become the de-facto standard for large network installations. In order to protect such installations, a novel approach for network-based intrusion detection is necessary to manage the ever-increasing data volume.

Network speeds have increased faster than the speed of processors, and therefore centralized solutions have reached their limit. This is especially true if one considers in-depth, stateful intrusion detection analysis. In this case, the sensors have to maintain information about attacks in progress (e.g., in the case of multi-step attacks) or they have to perform application-level analysis of the packet contents. These tasks are resource intensive and in a single-node setup may seriously interfere with the basic task of retrieving packets from the wire.

To be able to perform in-depth, stateful analysis it is necessary to divide the traffic volume into smaller portions that can be thoroughly analyzed by intrusion detection sensors. This approach has often been advocated by the high-performance research community as a way to distribute the service load across many nodes. In contrast to the case for standard load balancing, the division (or slicing) of the traffic for intrusion detection has to be performed in a way that guarantees the detection of all the threat scenarios considered. If a random division of traffic is used, sensors may not receive sufficient data to detect an intrusion, because different parts of the manifestation of an attack may have been assigned to different slices. Therefore, when an attack scenario consists of a number of steps, the slicing mechanism must assure that all of the packets that could trigger those steps are sent to the sensor configured to detect that specific attack.

#### A. A Slicing Approach to High-Speed Intrusion Detection

The problem of intrusion detection analysis in high-speed networks can be effectively attacked only if a scalable solution is available. Consider the traffic on a monitored network link as a bi-directional stream of link-layer frames (e.g., Ethernet frames). This stream contains too much data to be processed in real-time by a centralized entity and has to be divided into several smaller streams that are fed into a number of different, distributed sensors. Each sensor is only responsible for a subset of all detectable intrusion scenarios and can therefore manage to process the incoming volume in real-time. Nevertheless, the division into streams has to be done in a way that provides each sensor with enough information to detect exactly the same attacks that it would have witnessed when operating directly on a single network link.

#### B. Requirements

The overall goal is to perform stateful intrusion detection analysis in high-speed networks. The approach is characterized by the following requirements.

- The system must implement a misuse detection approach where *signatures* representing attack scenarios are matched against a stream of network events.

- Intrusion detection must be performed by a set of sensors, each of which is responsible for the detection of a subset of the signatures.
- Each sensor must be autonomous and must not interact with other sensors.
- The system must partition the analyzed event stream into slices of manageable size.
- Each traffic slice must be analyzed by a subset of the intrusion detection sensors.
- The system must guarantee that the partitioning of traffic maintains detection of all the specified attack scenarios. This implies that sensors, signatures, and traffic slices must be configured so that each sensor has access to the traffic necessary to detect the signatures that have been assigned to it.
- Components can be added to the system to achieve higher throughput. More precisely, the approach must result in a scalable design where one can add components as needed to match increased network throughput.

#### C. System Architecture

The requirements listed in the previous section have been used as the basis for the design of our network-based intrusion detection system. The system consists of a *network tap*, a *traffic scatterer*, a set of  $m$  *traffic slicers*  $S_0, \dots, S_{m-1}$ , a *switch*, a set of  $n$  *stream reassemblers*  $R_0, \dots, R_{n-1}$ , and a set of  $p$  *intrusion detection sensors*  $I_0, \dots, I_{p-1}$ . A high-level description of the architecture is shown in Figure 2.

The network tap component monitors the traffic stream on a high-speed link. Its task is to extract the sequence  $F$  of link-layer frames  $\langle f_0, f_1, \dots, f_t \rangle$  that are observable on the wire during a time period  $\Delta$ . This sequence of frames is passed to the scatterer which partitions  $F$  into  $m$  sub-sequences  $F_j : 0 \leq j < m$ . Each  $F_j$  contains a (possibly empty) subset of the frame sequence  $F$ . Every frame  $f_i$  is an element of exactly one sub-sequence  $F_j$  and therefore  $\cup_{j=0}^{m-1} F_j = F$ . The scatterer can use any algorithm to partition  $F$ . Hereafter, it is assumed that the scattering algorithm simply cycles over the  $m$  sub-sequences in a round-robin fashion, assigning  $f_i$  to  $F_{i \bmod(m)}$ . As a result, each  $F_j$  contains an  $m$ -th of the total traffic.

Each sub-sequence  $F_j$  is transmitted to a different traffic slicer  $S_j$ . The task of the traffic slicers is to route the frames they receive to the sensors that may need them to detect an attack. This task is not performed by the scatterer, because frame routing may be complex, requiring a substantial amount of time, while the scatterer has to keep up with the high traffic throughput and can only perform very limited processing per frame.

The traffic slicers are connected to a switch component, which allows a slicer to send a frame to one or more of  $n$  outgoing channels  $C_i$ . The set of frames sent to a channel is denoted by  $FC_i$ . Each channel  $C_i$  is associated with a stream reassembler component  $R_i$  and a number of intrusion detection sensors. The set of sensors associated with channel  $C_i$  is denoted by  $IC_i$ . All the sensors that are associated with a channel are able to access all the packets sent on that

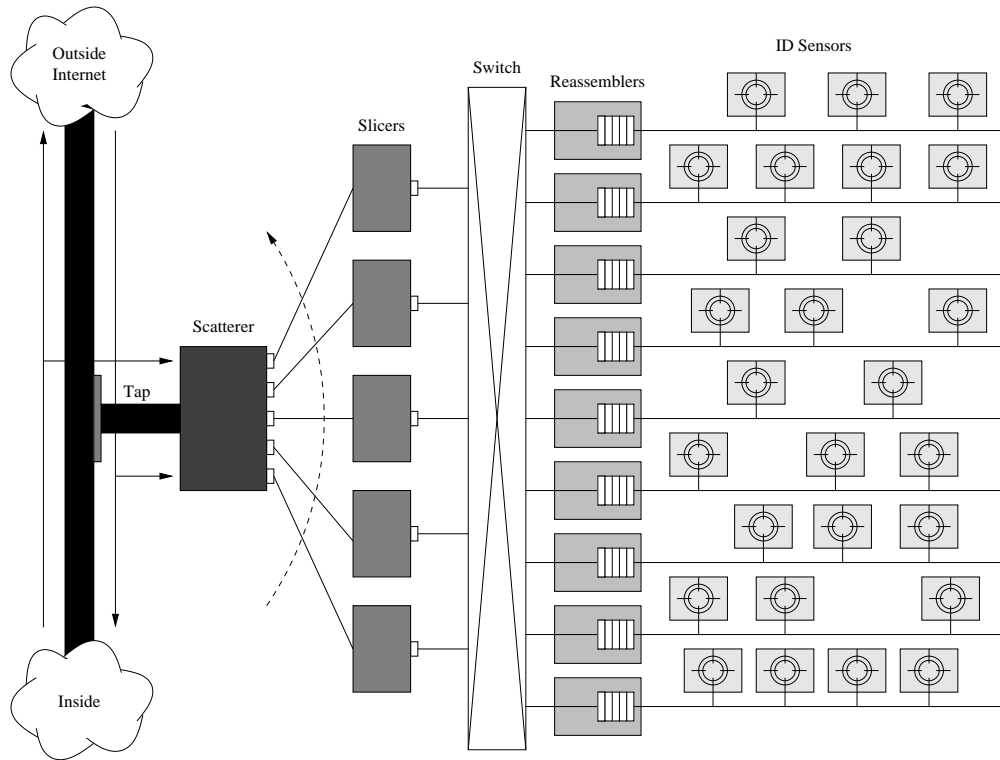


Fig. 2. High-level architecture of the high-speed intrusion detection system.

channel. A problem that could occur with this approach is that the original order of two packets could be lost if the two frames took different paths over distinct slicers to the same channel. Therefore, the reassemblers associated with each channel make sure that the packets appear on the channel in the same order that they appeared on the high-speed link. That is, each reassembler  $R_i$  must make sure that for each pair of frames  $f_j, f_k \in FC_i$ ,  $(f_j \text{ before } f_k) \iff j < k$ .

Each sensor component  $I_j$  is associated with  $q$  different attack scenarios  $A_j = \{A_{j0}, \dots, A_{jq-1}\}$ . Each attack scenario  $A_{jk}$  has an associated *event space*  $E_{jk}$ . The event space specifies which frames are candidates to be part of the manifestation of the attack. For example, consider an attack targeting a Web server called `georgia` within the network protected by the intrusion detection system. In this case, the event space for that attack is composed of all the TCP traffic that involves port 80 on host `georgia`.

Event spaces are expressed as disjunctions of *clauses*, that is,  $E_{jk} = c_{jk_0} \vee c_{jk_1} \vee \dots \vee c_{jk_n}$ , where each clause  $c_{jk}$  is an expression of the type  $xRy$ .  $x$  denotes a value derived from the frame  $f_i$  (e.g., a part of the frame header) while  $R$  specifies an arithmetic relation (e.g., =, !=, <).  $y$  can be a constant, the value of a variable, or a value derived from the same frame. Clauses and event spaces may be derived automatically from the attack descriptions, for example from signatures written in attack languages such as Bro [4], Sutekh [5], STATL [6], or Snort [7].

#### D. Frame Routing

Event spaces are the basis for the definition of the filters used by the slicers to route frames to different channels.

The filters are determined by composing the event spaces associated with all the scenarios that are “active” on a specific channel. More precisely, the set of active scenarios is  $AC_i = \bigcup_{j=0}^{j < k} A_j$  where  $A_j$  is the set of scenarios of  $I_j \in IC_i$  and  $k$  is the number of sensors on channel  $C_i$ . The event space  $EC_i$  for a channel  $C_i$  is the disjunction of the event spaces of all active scenarios, which corresponds to the disjunction of all the clauses of all the active scenarios. The resulting overall expression is the filter that each slicer uses to determine if a frame has to be routed to that specific channel. Note that it is possible that a certain frame will be needed by more than one scenario. Therefore, it will be sent on more than one channel.

The configuration of the slicers as described above is static; that is, it is calculated off-line before the system is started. The static approach suffers from the possibility that, depending on the type of traffic, a large percentage of the network packets could be forwarded to a single channel. This would result in an overloading of the sensors attached to that channel.

The static configuration also makes it impossible to predict the exact number of sensors that are necessary to deal with a Gigabit link. The load on each sensor depends on the scenarios used and the actual traffic. The minimum requirement for the slicers is that the capacity of their incoming and outgoing links must be at least equal to the bandwidth of the monitored link.

One way to prevent the overloading condition is to perform dynamic load balancing. This is done by reassigning scenarios to different channels at run-time. This variant obviously implies the need to reconfigure the filter mechanism at the traffic slicers and update the assignment of clauses to channels.

In addition to the reassignment of whole scenarios to different channels, it is also possible to split a single scenario

into two or more refined scenarios. The idea is that each refined scenario catches only a subset of the attacks that the original scenario covered, but each can be deployed on a different channel. Obviously, the union of attacks detectable by all refined scenarios has to cover exactly the same set of attacks as the original scenario did.

This can be done by creating additional constraints on certain attributes of one or more basic events. Each constraint limits the number of attacks a refined scenario can detect. The constraints have to be chosen in a way so that every possible value for a certain attribute (of the original scenario) is allowed by the constraint of at least one refined scenario. Then the set of all refined scenarios, which each cover only a subset of the attacks of the original one, are capable of detecting the same attacks as the original.

A simple mechanism to partition a particular scenario is to include a constraint on the destination attribute of each basic event that represents a packet which is sent by the attacker. One has to partition the set of possible destinations so that each refined scenario only covers attacks against a certain range of hosts. When the union of these target host ranges covers all possible attack targets, the set of refined scenarios is capable of finding the same attacks as the original scenario. This approach is necessary when a single scenario causes too much traffic to be forwarded to a single channel.

In addition, obviously innocent or hostile frames could be filtered out before the scenario clauses are applied, thereby eliminating traffic that needs no further processing. This could be used, for instance, to prevent the system from being flooded by packets from distributed denial-of-service slaves that produce traffic with a unique, known signature.

The effectiveness of the scatterer/slicer/reassembler architecture were experimentally evaluated on a system using three traffic slicers ( $m = 3$ ) and four stream reassemblers ( $n = 4$ ) with one intrusion detection sensor per stream with favorable results. The details of these experiments can be found in [8].

### III. HIGHLY CONFIGURABLE INTRUSION DETECTION SENSORS

Any monitoring and surveillance functionality builds on the analysis performed by *surveillance sensors*. The intrusion detection community has developed a number of different systems that perform intrusion detection in particular domains (e.g., hosts or networks) and in specific environments (e.g., Windows NT or Solaris).

These tools suffer from two main limitations: they are developed *ad hoc* for certain types of domains and/or environments, and they are difficult to configure, extend, and control remotely. In the specific case of signature-based intrusion detection systems [9]–[12] the sensors are equipped with a number of signatures that are matched against a stream of incoming events. Most systems (e.g., [9]) are initialized with a set of signatures at startup time. Updating the signature set requires stopping the sensor, updating the signature set, and then restarting execution. Some of these tools provide a way to enable/disable some of the available signatures, but few systems allow for the dynamic inclusion of new signatures at

execution time. In addition, the *ad hoc* nature of existing tools does not allow one to dynamically configure a running sensor so that a new event stream can be used as input for the security analysis.

Another limit of existing tools is the relatively static configuration of responses. As was the case for signatures, normally it is possible to choose only from a specific subset of possible responses. In addition, to our knowledge, no system allows for associating a response with *intermediate* steps of an attack. This is a severe limitation, especially in the case of distributed attacks carried out over a long time span.

In addition, the configuration of existing tools is mainly performed manually and at a very low level. This task is particularly error-prone, especially if the intrusion detection sensors are deployed across a very heterogeneous environment and with very different configurations. The challenge is to determine if the current configuration of one or more sensors is valid or if a reconfiguration is meaningful.

We have developed a novel approach to distributed intrusion detection. The idea is that a protected network is instrumented with a “web of sensors” composed of distributed components integrated by means of a local communication and control infrastructure. The task of the web of sensors is to provide fine-grained surveillance inside the protected network. The web of sensors implements *local surveillance* against both outside attacks and local misuse by insiders in a way that is complementary to the mainstream approach where a single point of access (e.g., a gateway) is monitored for possible malicious activity. The outputs of the sensors, in the form of *alerts*, are collected by a number of “meta-sensor” components. Each meta-sensor is responsible for a subset of the deployed sensors, and may coordinate its activities with other meta-sensors. The meta-sensors are responsible for storing the alerts, for routing alerts to other sensors and meta-sensors (e.g., to perform correlation to identify composite attack scenarios), and for exerting control over the managed sensors.

Control is the most challenging (and most overlooked) functionality of distributed surveillance. Most existing approaches simply aggregate the outputs of distributed sensors and focus mainly on the intuitive presentation of alerts to the network security officer. This is not enough. There is a need for fine-grained control of the deployed sensors in terms of scenarios to be detected, tailoring of the sensors with respect to the protected network, and dynamic control over the types of response. These are requirements that can be satisfied only if the surveillance sensors are *highly configurable* and if configuration can be performed dynamically, without stopping and restarting sensors when a reconfiguration is needed.

We have designed a suite of highly configurable surveillance sensors and a command and control meta-sensor that allows the network security officer to exert a very fine-grained control over the deployed surveillance infrastructure. Meta-sensors can be organized hierarchically to achieve scalability and can be replicated to support fault-tolerance. This web of sensors is built around the State Transition Analysis Technique (STAT) framework developed by the Reliable Software Group at UCSB. The STAT framework provides a platform for the development of highly configurable probes in different

domains and environments. The STAT approach is centered around five key concepts: the STAT technique, the STATL language, the STAT Core, the CommSTAT communication infrastructure, and the MetaSTAT control system.

The approach provides the basic mechanisms to reconfigure, at run-time, which input event streams are analyzed by each sensor, which scenarios have to be used for the analysis, and what types of responses must be carried out for each stage of the detection process. In addition, the approach explicitly models the dependencies among the modules composing a sensor so that it is possible to automatically identify the steps that are necessary to perform a reconfiguration of the deployed sensing infrastructure. In addition, the possibility of retrieving current configurations from remote sensors allows one to determine if a reconfiguration is valid or meaningful.

#### A. The STAT Framework

The STAT framework is the result of the evolution of the original STAT technique and its application to UNIX systems [13]–[15] into a general framework for the development of STAT-based intrusion detection sensors [16].

1) *The STAT Technique*: STAT is a technique for representing high-level descriptions of computer attacks. Attack scenarios are abstracted into *states*, which describe the security status of a system, and *transitions*, which model the evolution between states. By abstracting from the details of particular exploits and by modeling only the key events involved in an attack scenario, STAT is able to model entire classes of attacks with a single scenario, overcoming some of the limitations of plain signature-based misuse detection systems [17].

2) *The STATL Language*: STATL is an extendible language [18] that is used to represent STAT attack scenarios. The language defines the domain-independent features of the STAT technique. The STATL language can be extended to express the characteristics of a particular domain and environment. The extension process includes the definition of the set of *events* that are specific to the particular domain or environment being addressed and the definition of new *predicates* on those events. For example, to extend STATL to deal with events produced by the Apache Web browser one would define one or more events that represent entries in the application logs. In this case an event would have the fields *host*, *ident*, *authuser*, *date*, *request*, *status*, and *bytes* as defined by Apache's Common Log Format (CLF) [19]. After having defined new events it may be necessary to specify specific predicates on those events. For example, the predicate `isCGIrequest()` would return true if an event is a request for a CGI script. Event and predicate definitions are grouped in a *language extension*. Once the event set and associated predicates for a language extension are defined, it is possible to use them in a STATL scenario description by including them with the STATL use keyword. A number of extensions for TCP/IP networks, Sun BSM audit records [20], and Windows NT event logs have been developed.

STATL scenarios are matched against a stream of events by the STAT core (described below). In order to have a scenario processed by the STAT core it is necessary to compile it

into a *scenario plugin*, which is a shared library (e.g., a “.so” library in UNIX or a DLL library in Windows). In addition, each language extension used by the scenario must be compiled into an *extension module*, which is a shared library too. Both STATL scenarios and language extension are translated into C++ code and compiled into libraries by the STAT development tools.

3) *The STAT Core*: The STAT core is the runtime of the STATL language. The STAT core implements the domain-independent characteristics of STATL, such as the concepts of state, transition, timer, matching of events, etc. At run-time the STAT core performs the actual intrusion detection analysis process by matching an incoming stream of events against a number of scenario plugins. A running instance of the STAT core is dynamically extended to build a STAT-based sensor. The details of this process can be found in [21].

4) *The CommSTAT communication infrastructure*: STAT-based sensors are connected by a communication infrastructure that allows the sensors to exchange alert messages and control directives in a secure way. STAT control messages are used to manage and update the configuration of STAT-based sensors. For example, messages can be used to ship a scenario plugin to a remote sensor and have it loaded into the core. As another example, the infrastructure supports messages to manage language extensions and other modules. Participation in the CommSTAT communication infrastructure is mediated by a *CommSTAT proxy* that performs preprocessing of messages and control directives and that supports the integration of third-party tools that are not based on the STAT framework.

5) *The MetaSTAT control infrastructure*: The CommSTAT communication infrastructure is used by the MetaSTAT component to exert control over a set of sensors. The MetaSTAT component is responsible for the following tasks:

- **Collect and store the alerts produced by the managed sensors.** IDMEF alerts are stored in a relational database. A schema to efficiently store and retrieve IDMEF alerts has been developed, and a GUI for the querying and display of stored alerts has been implemented.
- **Route alerts to STAT sensors and other MetaSTAT instances.** MetaSTAT components and STAT-based sensors can subscribe for specific alerts. Alerts matching a subscription are routed through the appropriate CommSTAT communication channels.
- **Maintain a database of available modules and relative dependencies.** Each MetaSTAT component is associated with a *Module Database* of compiled scenario plugins, language extension modules, and other modules. For each module, the database stores the dependencies with respect to both other modules and the operational environment where the module may need to be deployed. These dependencies are a novel aspect of the STAT approach and are described in more detail in [21].
- **Maintain a database of current sensor configurations.** MetaSTAT manages a *Sensor Database* containing the current components that are active or installed at each STAT-based sensor. This “privileged” view of the deployed web of sensors is the basis for controlling the sensors and planning reconfigurations of the surveillance

infrastructure. The structure of the database is described in detail in [22].

The STAT-based framework has been leveraged to realize a highly-configurable “web of sensors” controlled by a meta-sensor component, called MetaSTAT. The flexibility of the framework allows the Intrusion Detection Administrator to perform complex reconfiguration tasks. In addition, by explicitly modeling the dependencies between modules it is possible to automatically generate a valid deployment plan from high-level specifications.

The “web of sensors” is based on the STAT approach but it has been designed to be open. Third party IDS modules can easily be integrated through CommSTAT proxies. Integration of external components is limited to the exchange of alerts if primitives for the dynamic configuration of sensors are not available.

The STAT tools [23]–[28] and the MetaSTAT infrastructure [29] have been used in a number of evaluation efforts, such as the MIT/Lincoln Labs evaluations and the Air Force Rome Labs evaluations [30], [31], in technology integration experiments, such as DARPA’s Grand Challenge Problem (GCP), and the iDemo technology integration effort [32]. In all of these very different settings, the STAT tools performed very well: detecting attacks in real-time with very limited overhead.

The STAT Framework, the MetaSTAT infrastructure, and the STAT-based tools are open-source and publicly available at the STAT web site <http://www.cs.ucsb.edu/~rsg/STAT>.

#### IV. INTRUSION DETECTION ALERT CORRELATION

The intrusion detection community has developed a number of different intrusion detection systems that perform intrusion detection in particular domains (e.g., hosts or networks), in specific environments (e.g., Windows NT or Solaris), and at different levels of abstraction (e.g., kernel-level tools or application-level tools). As more IDSs are developed, network security officers (NSOs) are faced with the task of analyzing an increasing number of alerts resulting from the analysis of different event streams. In addition, IDSs are far from perfect and may produce both false positives and non-relevant positives. Non-relevant positives are alerts that correctly identify an attack, but the attack fails to meet its objective. For instance, the attack may be exercising a vulnerability in a service that is not provided by the victim host. For instance, consider a “Code Red” worm that attacks a Linux Apache server. Although an actual attack is seen on the network, this attack will fail, because Apache is not vulnerable to the exploit utilized by the worm. Clearly, there is a need for tools and techniques that allow an NSO to aggregate and combine the outputs of multiple IDSs, filter out spurious or incorrect alerts (such as “Code Red” attacks against Apache installations), and provide a succinct, high-level view of the security state of the protected network.

To address this issue, researchers and vendors have proposed *alert correlation*, an analysis process that takes the alerts produced by intrusion detection systems and produces compact reports on the security status of the network under surveillance. Although a number of correlation approaches have been suggested, there is no consensus on what this process is or how

it should be implemented and evaluated. In particular, most existing correlation approaches operate on only a few aspects of the correlation process, such as the fusion of alerts that are generated by different intrusion detection systems in response to a single attack, or the identification of multi-step attacks that represent a sequence of actions performed by the same attacker. In addition, correlation tools that do cover multiple aspects of the correlation process are evaluated “as a whole,” without an assessment of the effectiveness of each component of the analysis process. As a result, it is not clear if and how the different parts of the correlation process contribute to the overall goals of correlation.

Another problem is that many existing approaches operate under the assumption that the alert stream is composed of detections of successful attacks. Unfortunately, experience shows that this assumption is wrong. Intrusion detection systems are very noisy, and, in addition to false positives, they produce alerts with different levels of relevance. As a consequence, the effectiveness of alert correlation is negatively affected by the poor quality of the input alert stream.

The main objective of a correlation process is to produce a succinct overview of security-related activity on the network. This process consists of a collection of components that transform intrusion detection sensor alerts into intrusion reports. Because alerts can refer to different kinds of attacks at different levels of granularity, the correlation process cannot treat all alerts equally. Instead, it is necessary to provide a set of components that focus on different aspects of the overall correlation task. We have designed and implemented a tool, called AlertSTAT, which uses the MetaSTAT infrastructure to collect the alerts produced by the intrusion detection sensors and then performs correlation on the resulting data using a number of different components.

Some of the components can operate on all alerts, independent of their type. These components are used in the initial and final phases of the correlation process. Other components work with only certain classes of alerts. These components are responsible for performing specific correlation tasks that cannot be generalized for arbitrary alerts.

The correlation tool that we implemented uses a pipeline architecture, consisting of 10 different components. The first two tasks are performed on all alerts. The process starts with a *normalization* component that translates every alert that is received into a standardized format that is understood by all other correlation components. This is necessary because alerts from different sensors can be encoded in different formats. Next, a *pre-processing* component augments the normalized alerts so that all required alert attributes (such as start-time, end-time, source, and target of the attack) are assigned meaningful values.

The core of the correlation process consists of components that implement specific functions, which operate on different spatial and temporal properties. That is, some of the components correlate events that occur close in time and space (e.g., alerts generated on one host within a small time window), while others operate on events that represent an attack scenario that evolves over several hours and that includes alerts that are generated on different hosts (e.g., alerts that represent large-

scale scanning activity). It is natural to combine events that are closely related (spatially and temporally) into composite alerts, which are in turn used to create higher-level alerts.

The next four correlation components of our process all operate on single, or closely related, events. The *fusion* component is responsible for combining alerts that represent the independent detection of the same attack instance by different intrusion detection systems. The *verification* component takes a single alert and determines the success of the attack that corresponds to this alert. The idea is that alerts that correspond to failed attacks should be appropriately tagged and their influence on the correlation process should be decreased. The *thread reconstruction* component combines a series of alerts that refer to attacks launched by a single attacker against a single target. The *attack session reconstruction* component associates network-based alerts with host-based alerts that are related to the same attack.

The next two components in our process operate on alerts that involve a potentially large number of different hosts. The *focus recognition* component has the task of identifying hosts that are either the source or the target of a substantial number of attacks. This is used to identify denial-of-service (DoS) attacks or port scanning attempts. The *multi-step correlation* component has the task of identifying common attack patterns, such as an island-hopping attack (i.e., an attack where an intruder breaks into a host and uses it as a launch pad for more attacks). These patterns are composed of a sequence of individual attacks, which can occur at different points in the network.

The final components of the correlation process contextualize the alerts with respect to a specific target network. The *impact analysis* component determines the impact of the detected attacks on the operation of the network being monitored and on the assets that are targeted by the malicious activity. The results of this analysis are then used by the *prioritization* component to assign an appropriate priority to every alert. This priority information is important for quickly discarding information that is irrelevant or of less importance to a particular site.

In its current configuration, alerts that are correlated by one AlertSTAT component are input to the next component. Although this process is sequential, all alerts are not required to pass through components sequentially. Some components could operate in parallel, and it is even possible that alerts output by a sequence of components could be fed back as input to a previous component of the process.

We have applied our correlation tool to a large number of diverse data sets, to analyze if and how each component contributes to the overall correlation process. The details of these experiments can be found in [33]. These experiments demonstrate that the effectiveness of each component of the process is dependent on the data sets being analyzed. To be more precise, these experiments show that the performance of the correlation process is significantly influenced by the network topology, the characteristics of the attack, and the available meta-data. Thus, one cannot, in general, determine a ranking among components with respect to their effectiveness.

A preliminary version of our correlation tool was one of

five systems that were independently evaluated in DARPA's Correlation Technology Validation effort [32]. Even though, at that time, the prototype included only a subset of the components described in this paper, it was ranked as one of the top correlators. The prototype produced two orders of magnitude reduction in the number of alerts and operated in real-time.

## V. RELATED WORK

Hi-DRA composes a number of different approaches to provide a comprehensive intrusion detection system for high-speed, wide-area networks. There is a large body of related research for each of the areas that these approaches represent. Unfortunately, due to space limitations, it is not feasible to present all the related work that has been published in these fields. Instead we refer the interested reader to the key papers that present the details of the three Hi-DRA components discussed in this paper for more information on related research. A discussion of most of the related research can be found in [18], [21], [22], [33].

It is possible, however, to compare our system to other multi-component, integrated systems that have been proposed by both the research and the commercial communities. Notable examples are SRI's EMERALD [10], [34] and ISS' RealSecure [12]. Both of these toolsets include a number of different sensor components and high-level analysis engines. For example, EMERALD includes a host-based intrusion detection system, two network-based analyzers, and a correlation/aggregation component. Even though the EMERALD toolset covers a number of different domains, there is no explicit mechanism in their approach that is exclusively dedicated to support the extension of an intrusion detection system to new domains. In addition, there is no infrastructure to control and reconfigure the sensors at run-time. RealSecure suffers from similar limitations. Being a commercial system, RealSecure does not provide an extensible framework for the creation of new sensors or for the tailoring of existing sensors to specific security requirements. Compared to these systems, Hi-DRA provides a finer-grained level of control and customizability.

## VI. CONCLUSIONS AND FUTURE WORK

This paper presented Hi-DRA, a system that integrates intrusion detection, alert correlation, and coordination mechanisms, to achieve in-depth surveillance in high-speed, wide-area networks. By composing localized monitoring with global communication and control, Hi-DRA is able to support in-depth analysis and high-level reporting and configuration.

Future work will focus on experimenting to improve two aspects of the system. First, we plan to refine our network models to better evaluate the impact that wide-spread attacks could have on a large-scale infrastructure. It is possible that these attacks could leverage complex, unforeseen interdependencies between subsystems that are difficult to represent and analyze. Secondly, we plan to use simulation-based analysis to evaluate the ability of the system to scale to hundreds of networks with thousands of sensors. These large-scale

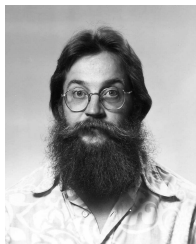
installations represent a challenge in terms of the resources needed to manage the volume of alerts produced by the sensors and to the ability to map high-level control directives onto a complex, heterogeneous infrastructure.

#### ACKNOWLEDGMENT

This research was supported by the Army Research Laboratory and the Army Research Office, under agreement DAAD19-01-1-0484. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

#### REFERENCES

- [1] G. Vigna, F. Valeur, J. Zhou, and R. Kemmerer, "Composable Tools For Network Discovery and Security Analysis," in *Proceedings of the 18<sup>th</sup> Annual Computer Security Applications Conference (ACSAC '02)*. Las Vegas, NV: IEEE Press, December 2002, pp. 14–24.
- [2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Achieving scalability and expressiveness in an internet-scale event notification service," in *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, Portland OR, USA, July 2000, pp. 219–227.
- [3] NSS Group, "Intrusion Detection and Vulnerability Assessment," NSS, Oakwood House, Wennington, Cambridgeshire, UK, Tech. Rep., 2000.
- [4] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," in *Proceedings of the 7<sup>th</sup> USENIX Security Symposium*, San Antonio, TX, January 1998.
- [5] J. Pouzol and M. Ducassé, "From Declarative Signatures to Misuse IDS," in *Proceedings of the RAID International Symposium*, ser. LNCS, W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Davis, CA: Springer-Verlag, October 2001, pp. 1 – 21.
- [6] S. Eckmann, G. Vigna, and R. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," in *Proceedings of the ACM Workshop on Intrusion Detection Systems*, Athens, Greece, November 2000.
- [7] M. Roesch, *Writing Snort Rules: How To write Snort rules and keep your sanity*, <http://www.snort.org>.
- [8] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful Intrusion Detection for High-Speed Networks," in *Proceedings of the IEEE Symposium on Security and Privacy*. Oakland, CA: IEEE Press, May 2002, pp. 285–293.
- [9] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of the USENIX LISA '99 Conference*, November 1999.
- [10] P. Neumann and P. Porras, "Experience with EMERALD to Date," in *First USENIX Workshop on Intrusion Detection and Network Monitoring*, Santa Clara, California, April 1999, pp. 73–80.
- [11] *Overview of NFR Network Intrusion Detection System*, NFR Security, February 2001.
- [12] ISS, "Realsecure," <http://www.iss.net/>, 2005.
- [13] K. Ilgun, "USTAT: A Real-time Intrusion Detection System for UNIX," Master's thesis, Computer Science Department, University of California, Santa Barbara, July 1992.
- [14] —, "USTAT: A Real-time Intrusion Detection System for UNIX," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 1993.
- [15] P. Porras, "STAT – A State Transition Analysis Tool for Intrusion Detection," Master's thesis, Computer Science Department, University of California, Santa Barbara, June 1992.
- [16] G. Vigna, S. Eckmann, and R. Kemmerer, "The STAT Tool Suite," in *Proceedings of DISCEX 2000*. Hilton Head, South Carolina: IEEE Computer Society Press, January 2000.
- [17] K. Ilgun, R. Kemmerer, and P. Porras, "State Transition Analysis: A Rule-Based Intrusion Detection System," *IEEE Transactions on Software Engineering*, vol. 21, no. 3, pp. 181–199, March 1995.
- [18] S. Eckmann, G. Vigna, and R. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," *Journal of Computer Security*, vol. 10, no. 1/2, pp. 71–104, 2002.
- [19] *Apache 2.0 Documentation*, 2002, <http://www.apache.org/>.
- [20] Sun Microsystems, Inc., *Installing, Administering, and Using the Basic Security Module*, 2550 Garcia Ave., Mountain View, CA 94043, December 1991.
- [21] G. Vigna, F. Valeur, and R. Kemmerer, "Designing and Implementing a Family of Intrusion Detection Systems," in *Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003)*, Helsinki, Finland, September 2003.
- [22] G. Vigna, R. Kemmerer, and P. Blix, "Designing a Web of Highly-Configurable Intrusion Detection Sensors," in *Proceedings of the 4<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, ser. LNCS, W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Davis, CA: Springer-Verlag, October 2001, pp. 69–84.
- [23] G. Vigna and R. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach," in *Proceedings of the 14<sup>th</sup> Annual Computer Security Application Conference*, Scottsdale, Arizona, December 1998.
- [24] —, "NetSTAT: A Network-based Intrusion Detection System," *Journal of Computer Security*, vol. 7, no. 1, pp. 37–71, 1999.
- [25] G. Vigna, B. Cassell, and D. Fayram, "An Intrusion Detection System for Aglets," in *Proceedings of the 6<sup>th</sup> International Conference on Mobile Agents (MA '02)*, ser. LNCS, N. Suri, Ed., vol. 2535. Barcelona, Spain: Springer-Verlag, October 2002, pp. 64–77.
- [26] G. Vigna, W. Robertson, V. Kher, and R. Kemmerer, "A Stateful Intrusion Detection System for World-Wide Web Servers," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003)*, Las Vegas, NV, December 2003, pp. 34–43.
- [27] S. Soman, C. Krintz, and G. Vigna, "Detecting Malicious Java Code Using Virtual Machine Auditing," in *Proceedings of 12<sup>th</sup> USENIX Security Symposium*, V. Paxson, Ed. Washington, DC: USENIX, August 2003, pp. 153–167.
- [28] G. Vigna, S. Gwalani, K. Srinivasan, E. Belding-Royer, and R. Kemmerer, "An Intrusion Detection Tool for AODV-based Ad Hoc Wireless Networks," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, Tucson, AZ, December 2004, pp. 16–27.
- [29] R. Kemmerer and G. Vigna, "Sensor Families for Intrusion Detection Infrastructures," in *Managing Cyber Threats: Issues, Approaches and Challenges*. Kluwer Academic Publishers, 2004.
- [30] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo, "Testing and Evaluating Computer Intrusion Detection Systems," *CACM*, vol. 42, no. 7, pp. 53–61, July 1999.
- [31] —, "Addendum to "Testing and Evaluating Computer Intrusion Detection Systems"," *CACM*, vol. 42, no. 9, p. 15, September 1999.
- [32] J. Haines, D. Ryder, L. Tinnel, and S. Taylor, "Validation of Sensor Alert Correlators," *IEEE Security & Privacy Magazine*, vol. 1, no. 1, pp. 46–56, January/February 2003.
- [33] F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer, "A Comprehensive Approach to Intrusion Detection Alert Correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146–169, July–September 2004.
- [34] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in *Proceedings of the 1997 National Information Systems Security Conference*, October 1997.



**Richard A. Kemmerer** is a Professor and a past Chair of the Department of Computer Science at the University of California, Santa Barbara. Dr. Kemmerer received the B.S. degree in Mathematics from the Pennsylvania State University in 1966, and the M.S. and Ph.D. degrees in Computer Science from the University of California, Los Angeles, in 1976 and 1979, respectively. His research interests include formal specification and verification of systems, computer system security and reliability, programming and specification language design, and software engineering. He is author of the book *Formal Specification and Verification of an Operating System Security Kernel* and a co-author of *Computers at Risk: Safe Computing in the Information Age*, *For the Record: Protecting Electronic Health Information*, and *Realizing the Potential of C4I: Fundamental Challenges*. Dr. Kemmerer is a Fellow of the IEEE Computer Society, a Fellow of the Association for Computing Machinery, a member of the IFIP Working Group 11.3 on Database Security, and a member of the International Association for Cryptologic Research. He is a past Editor-in-Chief of *IEEE Transactions on Software Engineering* and has served on the editorial boards of the *ACM Computing Surveys* and *IEEE Security and Privacy*. He currently serves on the Board of Governors of the IEEE Computer Society and on Microsoft's Trustworthy Computing Academic Advisory Board.



**Giovanni Vigna** is an Associate Professor in the Department of Computer Science at the University of California in Santa Barbara. His current research interests include network and computer security, intrusion detection, security of mobile code systems, penetration testing, and wireless systems. In particular, he worked on STAT, a framework for the modular development of intrusion detection systems. He also published a book on Security and Mobile Agents and he has been the Program Chair of the International Symposium on Recent Advances in Intrusion Detection (RAID 2003). Giovanni Vigna received his M.S. with honors and Ph.D. from Politecnico di Milano, Italy, in 1994 and 1998, respectively.