Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 7 (2007) 631–641
Applied Soft Computing
www.elsevier.com/locate/asoc

Hybrid multi-agent framework for detection of stealthy probes

Srinivas Mukkamala [a,b,*], Andrew H. Sung [a,b], Ajith Abraham [c]

[a] Department of Computer Science, New Mexico Tech, USA
[b] Institute for Complex Additive Systems Analysis, New Mexico Tech, USA
[c] Department of Computer Science, Oklahoma State University, USA

Received 13 April 2004; accepted 2 February 2005
Available online 25 January 2006

Abstract

Probing tools are widely used to discover system information. Once the information is known, attackers can launch computer attacks against the vulnerable services running on the system. Even though current computer systems are protected against known attacks by implementing a number of access restriction policies, protection against novel attacks still remains as an elusive goal for the researchers. Attackers defeat current protection and detection mechanisms by exploiting unknown weakness and bugs in system and application software. Stealthy and low profile probes that include only a few carefully crafted packets over an extended period of time are used to delude firewalls and intrusion detection systems (IDS).

Building effective IDSs, unfortunately, has remained an elusive goal owing to the great technical challenges involved and applied AI techniques are increasingly being utilized in attempts to overcome the difficulties. This paper presents computational intelligent agents-based approach to detect computer probes at the originating host. We also investigate and compare the performance of different classifiers used for detecting probes, with respect to the data collected on a real network that includes a variety of simulated probe attacks and the normal activity.

Through a variety of experiments and analysis, it is found that with appropriately chosen network features computer probes can be detected in real time or near real time at the originating host. Using the detection information, an effective response mechanism can be implemented at the boundary controllers.

Keywords: Intrusion detection systems; Stealthy probes; Controller

1. Introduction

Intrusion detection is a problem of great significance to protecting information systems security, especially in view of the worldwide increasing incidents of cyber attacks on the critical infrastructures. Since the ability of an IDS to classify a large variety of intrusions in real time or in near real time with accurate results is important, we will consider performance measures in critical aspects like training and testing times; scalability; and classification accuracy.

One of the main problems with IDSs is the overhead, which can become unacceptably high. To analyze system logs, the operating system must keep information regarding all the actions performed, which invariably results in huge amounts of data, requiring disk space and CPU resource. Next, the logs must be processed and converted into a manageable format and then compared with the set of recognized misuse and attack patterns to identify possible security violations. Further, the stored patterns need be continually updated, which would normally involve human expertise. Constructing probing signatures is a bit more complex as the attackers use carefully crafted packets over a period of time that appear to be like normal network traffic. Traditional port scan detectors look for packets coming from several IP addresses with combination of different ports with in a defined time to a target machine, which involves computation to analyze packet headers to search for signatures. In this paper, we present a novel approach of detecting probes at the originating machine using computational intelligent agent-based techniques that reduces the computation at the server and efficiently deal with distributed attacks.

Several artificial intelligence techniques have been utilized to automate the intrusion detection process to reduce human intervention; several such techniques include neural networks [1–6], fuzzy inference systems, evolutionary computation machine learning [7,8], and so on. Several data mining techniques have been introduced to identify key features or

* Corresponding author at. Institute for Complex Additive Systems Analysis, New Mexico Tech, USA. Tel.: +1 505 835 5036; fax: +1 505 835 5587.
E-mail address: srinivas@cs.nmt.edu (S. Mukkamala).

1568-4946/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2005.12.002

parameters that define intrusions [9–12]. A summary of intrusion detection techniques is given by several research works [13,14]. Some works applied neural networks as classifiers to detect low level probes and a summary of different port-scan detection techniques is also available [15,16].

In this paper, we implement and evaluate the performance of computational intelligent multi-agent system to detect computer probes at the originating host. Intelligent agents were encapsulated with different AI paradigms involving support vector machines (SVM), multi-variate adaptive regression splines (MARS) and linear genetic programming (LGP) for detecting probes. Performance metrics include critical aspects of intrusion detection such as scalability, real time detection and accuracy. The IDS metric of real-time performance capability is not considered directly since it depends on the actual implementation. The data we use in our experiments is collected on a real performing network at New Mexico Technology, USA that includes normal activity and several classes of probing attacks generated using an open source tool Network Mapper (Nmap). We perform experiments to classify the network traffic sessions into "Normal" and "Probe". With appropriately chosen population size, program size, crossover rate and mutation rate, linear genetic programs outperform other artificial intelligent techniques in terms of detection accuracy. The experimental results of overall classification accuracy- and class-specific accuracies using SVM, MARS and LGP are reported.

A brief introduction to our computational intelligent agents-based architecture is given in Section 2. Data generation and collection is described in Section 3 of this paper. Section 3 also briefly explains the different AI paradigms we used for classifying normal activity and probes. In Section 4, we briefly describe the offline data analysis and feature extraction for real time detection of probes at the originating host. Real-time data collection and feature extraction are described in Section 5. Implementation details of the computationally intelligent multi-agent system are given in Section 6. Experimental results of using SVM, MARS and LGP as classifiers are given in Section 7. The summary and conclusions of our work are given in Section 8.

2. Computational intelligent agents (CIA)-based architecture

The CIA-based architecture for detecting computer attacks consists of several modules that will be executed by the agents in a distributed manner. Communication among the agents is done utilizing the TCP/IP sockets. Agent modules running on the host computers consist of data collection agents, data analysis agents and response agents. Agents running on the secure devices consist of the agent control modules that include agent regeneration, agent dispatch, maintaining intrusion signatures and information of the features and feature ranking algorithms that help identify the intrusions (Fig. 1).

2.1. Host agents

Reside on the hosts of the internal network and perform the tasks specified by the master agent. These agents are implemented to be read only and fragile. In the event of tampering or modification, the agent reports to the server agent and automatically ends its life (Fig. 2).

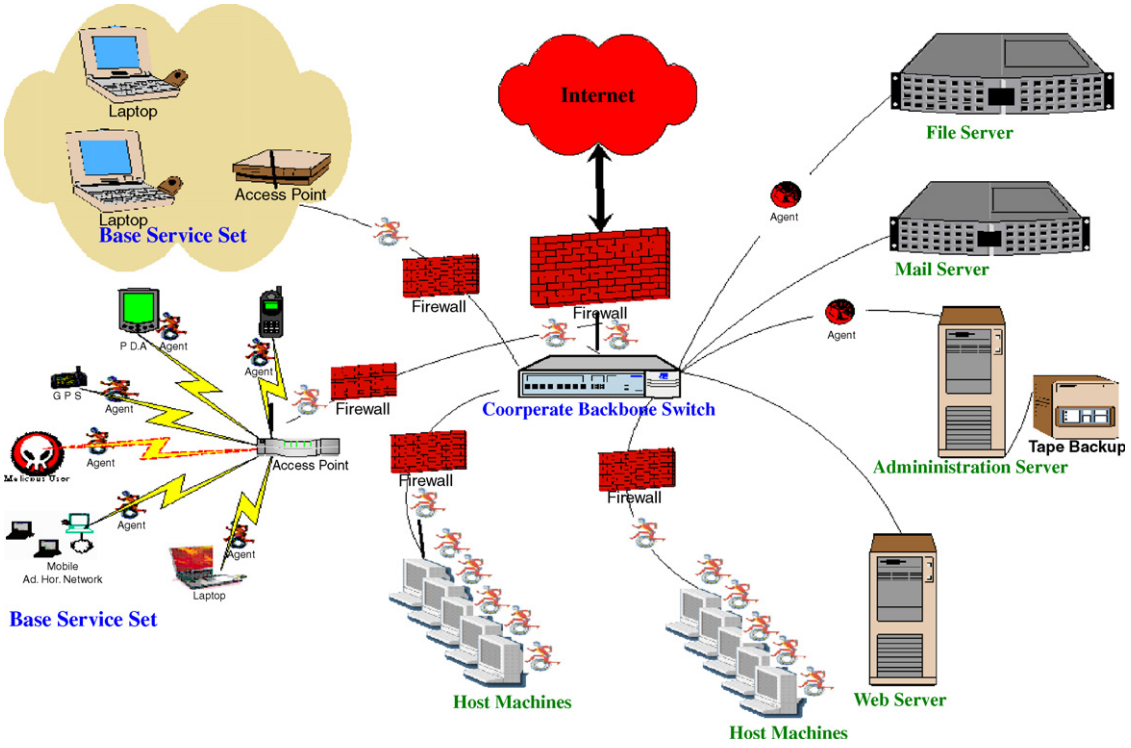

Fig. 1. Computational intelligent agents architecture [17].

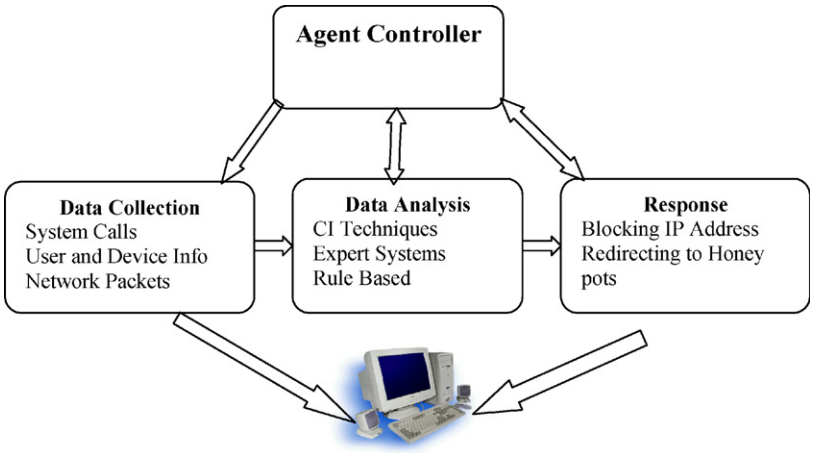

Fig. 2. Functionality of host-based agents.

2.2. Server agents

Reside on the secure server of the network. Controls the individual host agents for monitoring the network and manages communication between the agents if necessary. These agents manage the life cycle and also update the host agents with new detection, feature extraction, response and trace mechanisms (Fig. 3).

The two diagrams above contain a variety of modules, as described below:

- *Agent controller:* Manages the agents' functionality, life cycle, communication with other agents and the response mechanisms.
- *Attack signatures module:* Maintains all the attack signatures and updates monitoring agents in the event of a new attack signature.
- *Data collection:* Extracts features required by the detection algorithm to decide whether the activity is malicious or normal.
- *Intrusion detection module:* Consists of soft computing, rule based and expert systems for classifying intrusions. Includes a decision module that decides which classification algorithm should be used depending on the information collected by the monitoring agent.
- *Response module:* Decides weather a proactive (honey pots, decoys, traps) or a reactive (block the activity at the boundary) response mechanism should be implemented depending on the intensity of the intrusion.
- *Trace back initiation:* Initiates trace back with the help of the controller agent and the device they are residing on. In the event of a network attack, they use the information of the router tables and try to reach the boundary router of the attackers source by verifying the path at each boundary.

Advantages of the proposed model:

- with prior knowledge of the device and user profiles of the network, specific agents can be designed and implemented in a distributed fashion;
- highly optimized parallel algorithms can be designed and implemented independently for data collection agents, data analysis agents and response agents;
- analysis for computationally limited devices can be offloaded to cooperating systems and run in parallel to provide capabilities not otherwise available on such devices;

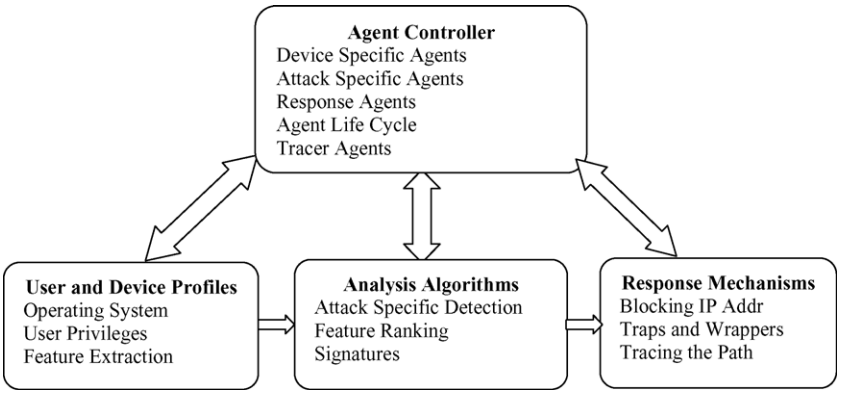

Fig. 3. Functionality of server agents.

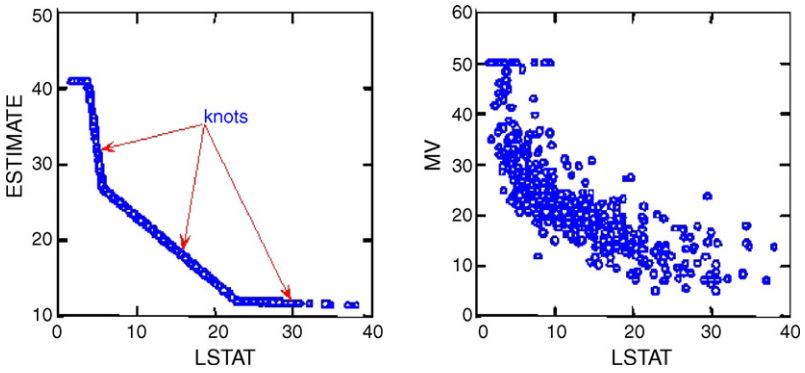

Fig. 4. MARS data estimation using spines and knots (actual data on the right) [20].

- attack-specific agents can be implemented and dispatched to respond to specific new threats;
- efficient detection algorithms can be implemented with less overhead;
- rapid intrusion response and trace back can be performed more easily with the agents communicating with each other;
- adjustable detection thresholds can be implemented.

3. Data mining: a computational intelligence approach

Data mining (also known as knowledge discovery in databases, KDD) has been defined by Frawley as ''The nontrivial extraction of implicit, previously unknown and potentially useful information from data''. Data mining techniques use machine learning, statistical and visualization techniques to discover and present knowledge from the raw information in a easily comprehensible form to humans. In the field of intrusion detection data mining programs are used to analyze audit trails and provide knowledge (features) that help in distinguishing intrusive patterns from normal activity [9–12].

It is well known that the intelligent systems, which can provide human like expertise such as domain knowledge, uncertain reasoning and adaptation to a noisy and time varying environment, are important in tackling practical computing problems. To detect stealthy probes, we investigated the classification performance of support vector machines (SVM), multi-variate adaptive regression splines (MARS) and linear genetic programming (LGP).

3.1. Multi-variative adaptive regression splines

Splines can be considered as an innovative mathematical process for complicated curve drawings and function approximation. To develop a spline, the x-axis is broken into a convenient number of regions. The boundary between regions is also known as a knot. With a sufficiently large number of knots virtually any shape can be well approximated. While it is easy to draw a spline in two-dimensions by keying on knot locations (approximating using linear, quadratic or cubic polynomial, etc.), manipulating the mathematics in higher dimensions is best accomplished using basis functions. The MARS model is a regression model using basis functions as predictors in place of the original data. The basis function transform makes it possible to selectively blank out certain regions of a variable by making them zero, and allows MARS to focus on specific sub-regions of the data. It excels at finding optimal variable transformations and interactions, and the complex data structure that often hides in high-dimensional data [18,19] (Fig. 4).

Given the number of records in most data sets, it is infeasible to approximate the function $y = f(x)$ by summarizing y in each distinct region of x. For some variables, two regions may not be enough to track the specifics of the function. If the relationship of y to some x's is different in three or four regions, for example, the number of regions requiring examination is even larger than 34 billion with only 35 variables. Given that the number of regions cannot be specified a priori, specifying too few regions in advance can have serious implications for the final model. A solution is needed that accomplishes the following two criteria:

- judicious selection of which regions to look at and their boundaries;
- judicious determination of how many intervals are needed for each variable.

Given these two criteria, a successful method will essentially need to be adaptive to the characteristics of the data. Such a solution will probably ignore quite a few variables (affecting variable selection) and will take into account only a few variables at a time (also reducing the number of regions). Even if the method selects 30 variables for the model, it will not look at all 30 simultaneously. Such simplification is accomplished by a decision tree at a single node, only ancestor splits are being considered; thus, at a depth of six levels in the tree, only six variables are being used to define the node [19].

3.2. Support vector machines (SVMs)

The SVM approach transforms data into a feature space F that usually has a huge dimension. It is interesting to note that SVM generalization depends on the geometrical characteristics of the training data, not on the dimensions of the input space [21,22]. Training a support vector machine (SVM) leads to a quadratic optimization problem with bound constraints and one linear equality constraint. Vapnik shows how training a SVM

for the pattern recognition problem leads to the following quadratic optimization problem [23].

$$\text{Minimize}: W(\alpha) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(x_i, x_j) \quad (1)$$

$$\text{Subject to} \sum_{i=1 \forall i: 0 \le \alpha_i \le C}^{l} y_i \alpha_i \quad (2)$$

where l is the number of training examples α is a vector of l variables and each component α_i corresponds to a training example (x_i, y_i). The solution of (1) is the vector α^* for which (1) is minimized and (2) is fulfilled.

3.3. Linear genetic programming (LGP)

LGP is a variant of the genetic programming (GP) technique that acts on linear genomes [24]. The linear genetic programming technique used for our current experiment is based on machine code level manipulation and evaluation of programs. Its main characteristics in comparison to tree-based GP lies is that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like C) are evolved. In the automatic induction of machine code by genetic programming, individuals are manipulated directly as binary code in memory and executed directly without passing an interpreter during fitness calculation. The LGP tournament selection procedure puts the lowest selection pressure on the individuals by allowing only two individuals to participate in a tournament. A copy of the winner replaces the loser of each tournament. The crossover points only occur between instructions. Inside instructions the mutation operation randomly replaces the instruction identifier, a variable or the constant from valid ranges. In LGP, the maximum size of the program is usually restricted to prevent programs without bounds. As LGP could be implemented at machine code level, it will be fast to detect intrusions in a near real time mode [24].

4. Offline feature extraction and evaluation

A sub set of the DARPA intrusion detection data set is used for offline analysis. In the DARPA intrusion detection evaluation program, an environment was set up to acquire raw TCP/IP dump data for a network by simulating a typical U.S. Air Force LAN. The LAN was operated like a real environment, but being blasted with multiple attacks [25,26]. For each TCP/IP connection, 41 various quantitative and qualitative features were extracted [9]. The 41 features extracted fall into three categories, ''intrinsic'' features that describe about the individual TCP/IP connections can be obtained from network audit trails, ''content-based'' features that describe about payload of the network packet can be obtained from the data portion of the network packet, ''traffic-based'' features, that are computed using a specific window (connection time or no of connections). As DOS and probe attacks involve several connections in a short time frame, whereas R2U and U2Su attacks are embedded in the data portions of the connection and often involve just a single connection; ''traffic-based'' features play an important role in deciding whether a particular network activity is engaged in probing or not.

Table 1
Probe attacks

Attack type	Service	Mechanism	Effect of the attack
Ipsweep	ICMP	Abuse of feature	Identifies active machines
Mscan	Multiple	Abuse of feature	Looks for known vulnerabilities
Nmap	Multiple	Abuse of feature	Identifies active ports on a machine
Saint	Multiple	Abuse of feature	Looks for known vulnerabilities
Satan	Multiple	Abuse of feature	Looks for known vulnerabilities

4.1. Probing

Probing is a class of attacks where an attacker scans a network or a host to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network or a host can use this information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and a precursor for automated attacks; requires very little technical expertise (Table 1):

Ipsweep: Probing attack performed against all operating systems using ICMP service where an adversary performs a surveillance sweep to determine which hosts are responding on a network. Information obtained from surveillance is useful to an adversary in launching automated attacks or in making the vulnerable hosts as stepping stones for future distributed attacks. This attack helps the adversary identify active machines on the network and might degrade services for legitimate users. Looking for multiple ping requests, destined for all possible machines on a network, all coming from the same host can help detect this attack.
Mscan: Probing tool used to perform an attack against all operating systems using multiple services; where an adversary uses both DNS zone transfers and/or brute force scanning of IP addresses to locate machines, and look for vulnerabilities to launch future attacks. This attack helps the adversary identify known vulnerabilities on the network and the host machine. Looking for connection requests from an out side machine to vulnerable services (netbios-ns, epmap, ms-sql-m, dameware, microsoft-ds, realsecure, domain, bind, imap, pop, NFS, cgi-bin, open X servers) within a specified period of time, can help detect this attack.
Nmap: General-purpose probing tool used to perform network scans against all operating systems using multiple services with user specified time intervals; an adversary can specify which services to scan for, how much time to wait between each service, and whether the services should be scanned sequentially or in a random order. This attack helps

the adversary identify services running, operating system, and known vulnerabilities on the network and the target machine. Looking for connection requests to multiple services within a specific time window can help detect this attack.

Saint: Security Administrator's Integrated Network Tool is used to gather information about remote hosts (all operating systems) using multiple services; an adversary uses a few network services such as finger, ftp, tftp, statd, rpc, NIS, NFS and other relevant network services. This attack helps the adversary identify network services running, system flaws, critical security flaws on the victims' machine. Looking for connections requests to specific network services from a machine other than an authorized machine within in a specific time window can help detect this attack.

Satan: Probing tool used to perform scans against all operating systems using a few network services; where an adversary uses legitimate network services to gather information on particular vulnerabilities on the victims' machine. Looking for connections requests to specific vulnerable network services from a machine other than an authorized machine within in a specific time window can help detect this attack.

4.2. Input feature selection

Feature selection is an important issue in intrusion detection. Of the large number of features that can be monitored for intrusion detection purpose, which are truly useful, which are less significant, and which may be useless? The question is relevant because the elimination of useless features (the so-called audit trail reduction) enhances the accuracy of detection while speeding up the computation, thus improving the overall performance of IDS. In cases where there are no useless features, by concentrating on the most important ones we may well improve the time performance of an IDS without affecting the accuracy of detection in statistically significant ways.

The feature selection problem for intrusion detection is similar in nature to various engineering problems that are characterized by:

- Having a large number of input variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ of varying degrees of importance to the output $\boldsymbol{y}$; i.e., some elements of $\boldsymbol{x}$ are essential, some are less important, some of them may not be mutually independent, and some may be useless or irrelevant (in determining the value of $\boldsymbol{y}$).
- Lacking an analytical model that provides the basis for a mathematical formula that precisely describes the input–output relationship, $y = F(x)$.
- Having available a finite set of experimental data, based on which a model (e.g., intelligent systems) can be built for simulation and prediction purposes.

Due to the lack of an analytical model, one can only seek to determine the relative importance of the input variables through empirical methods. A complete analysis would require examination of all possibilities, e.g., taking two variables at a time to analyze their dependence or correlation, then taking three at a time, etc. This, however, is both infeasible (requiring 2^n experiments!) and not infallible (since the available data may be of poor quality in sampling the whole input space). Features are ranked based on their influence towards the final classification. Description of most important features as ranked by three feature-ranking algorithms (SVDF, LGP, MARS) is given in Table 2. The

Table 2
Most important feature description as ranked by (SVDF, LGP, MARS)

Ranking algorithm	Feature description
SVDF	Source bytes: number of bytes sent from the host system to the destination system dst_host_srv_count:: number of connections from the same host with same service to the destination host during a specified time window Count: number of connections made to the same host system in a given interval of time Protocol type: type of protocol used to connect (e.g., tcp, udp, icmp, etc.) srv_count: number of connections to the same service as the current connection during a specified time window Flag: normal or error status of the connection
LGP	dst_host_diff_srv_rate: % of connections to different services from a destination host rerror_rate: % of connections that have REJ errors srv_diff_host_rate: % of connections that have same service to different hosts Logged in: binary decision Service: type of service used to connect (e.g., fingure, ftp, telnet, ssh, etc.) Source bytes: number of bytes sent from the host system to the destination system
MARS	dst_host_diff_srv_rate: % of connections to different services from a destination host dst_host_srv_count:: number of connections from the same host with same service to the destination host during a specified time window Source bytes: number of bytes sent from the host system to the destination system dst_host_same_srv_rate: % of connections to same service ports from a destination host srv_count:: number of connections to the same service as the current connection during a specified time window rerror_rate: % of connections that have REJ errors

Table 3
Classifier performance of all 41 features and most important 6 features

Class	Normal (41 features)	Probe (41 features)	Normal (6 features)	Probe (6 features)
SVM	99.55	99.70	99.23	99.16
MARS	96.08	92.32	94.34	90.79
LGP	99.89	99.85	99.77	99.87

(training and testing) data set contains 11,982 randomly generated points from the five classes, with the number of data from each class proportional to its size, except that the smallest class is completely included. The normal data belongs to class1, probe belongs to class 2, denial of service belongs to class 3, user to super user belongs to class 4 and remote to local belongs to class 5. Where attack data is a collection of 22 different types of attack instances that belong to the four classes probe, denial of service, user to super user and remote to local. A different randomly selected set of 6890 points of the total data set (11,982) is used for testing different intelligent techniques. Classifier performance using all the 41 features and most important 6 features as inputs to the classifier is given in Table 3.

4.3. SVM-specific feature ranking method

Information about the features and their contribution towards classification is hidden in the support vector decision function. Using this information one can rank their significance, i.e., in the equation

$$F(X) = \sum W_i X_i + b$$

The point X belongs to the positive class if $F(X)$ is a positive value. The point X belongs to the negative class if $F(X)$ is negative. The value of $F(X)$ depends on the contribution of each value of X and W_i. The absolute value of W_i measures the strength of the classification. If W_i is a large positive value then the ith feature is a key factor for positive class. If W_i is a large negative value then the ith feature is a key factor for negative class. If W_i is a value close to zero on either the positive or the negative side, then the ith feature does not contribute significantly to the classification. Based on this idea, a ranking can be done by considering the support vector decision function.

4.3.1. Support vector decision function ranking

The input ranking is done as follows: First the original data set is used for the training of the classifier. Then the classifier's decision function is used to rank the importance of the features. The procedure is:

1. Calculate the weights from the support vector decision function;
2. Rank the importance of the features by the absolute values of the weights.

4.4. Ranking algorithm using evolutionary algorithms

The performance of each of the selected input feature subsets is measured by invoking a fitness function with the correspondingly reduced feature space and training set and evaluating the intrusion detection accuracy. Once the required number of iterations are completed, the evolved high ranked programs are analyzed for how many times each input appears in a way that contributes to the fitness of the programs that contain them. The best feature subset found is then output as the recommended set of features to be used in the actual input for the intrusion detection model [27].

In the feature selection problem, the main interest is in the representation of the space of all possible subsets of the given input feature set. Each feature in the candidate feature set is considered as a binary gene and each individual consists of fixed-length binary string representing some subset of the given feature set. An individual of length d corresponds to a d-dimensional binary feature vector Y, where each bit represents the elimination or inclusion of the associated feature. Then, $y_i = 0$ represents elimination and $y_i = 1$ indicates inclusion of the ith feature. Fitness F of an individual program p is calculated as the mean square erro (MSE) between the predicted output (O_{ij}^{pred}) and the desired output (O_{ij}^{des}) for all n training samples and m outputs [24].

$$F(p) = \frac{1}{n \cdot m}\sum_{i=1}^{n}\sum_{j=1}^{m}(O_{\text{ij}}^{\text{pred}} - O_{\text{ij}}^{\text{des}})^2 + \frac{w}{n}\text{CE}$$

$$= \text{MSE} + w \cdot \text{MCE}$$

Classification error (CE) is computed as the number of misclassifications. Mean classification error (MCE) is added to the fitness function while its contribution is proscribed by an absolute value of weight (W).

4.5. Ranking algorithm using MARS

Generalized cross-validation is an estimate of the actual cross-validation which involves more computationally intensive goodness of fit measures. Along with the MARS procedure, a generalized cross-validation procedure is used to determine the significant input features. Non-contributing input variables are thereby eliminated.

$$\text{GCV} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - f(x_i)^2}{1 - k/N}\right]$$

where N is the number of records and x and y are independent and dependent variables, respectively. k is the effective number of degrees of freedom whereby the GCV adds penalty for adding more input variables to the model. The contribution of the input variables may be ranked using the GCV with/without an input feature [19,20].

5. Real-time data collection and feature extraction

Experiments are performed on a real network using two clients and the server that serves the New Mexico Tech Computer Science Department network. The clients had CIA installed on them to identify or detect probes that are targeted to the server we are protecting. Our primary goal in these experiments is to detect probes targeting the server we are trying to protect. Our network parser gives the summary of each connection made from a host to the server and constructs a feature set to input into a classifier for classification. The output from a classifier is either normal or probe for each connection. Nmap an open source tool is used to collect probe data [28]. Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. There are different types of probes: some of them abuse the computer's legitimate features; some of them use social engineering techniques. This class of attacks is the most commonly heard and requires very little technical expertise. Nmap is installed on the clients that have CIA installed. A variety of probes SYN stealth, FIN stealth, ping sweep, UDP scan, null scan, xmas tree, IP scan, idle scan, ACK scan, window scan, RCP scan and list scan with several options are targeted at the server. Table 4 describes the probing attacks performed on a live performance network. Normal data included multiple sessions of ftp, telnet, SSH, http, SMTP, pop3 and imap. Network data originating from a host to the server that included both normal and probes is collected for analysis; for proper labeling of data for training the classifiers normal data and probe data are collected at different times:

SYN stealth scan: Probing attack performed against all operating systems using multiple TCP services where an adversary performs surveillance to determine which hosts are responding to specific services on a network. Information obtained from surveillance is useful to an adversary in launching automated attacks, in making the vulnerable hosts as stepping stones for future distributed attacks or for launching future denial of service attacks. This attack helps the adversary identify active machines on the network and might degrade services for legitimate users. Looking for multiple half open TCP connection requests, destined for all possible machines on a network, can help detect this attack.
FIN stealth scan: Probing attack performed against all operating systems except Windows 95/NT; when SYN scanning isn't clandestine enough. By theory closed ports are required to reply to a probe packet with a RST, while open ports must ignore the packets. An adversary abuses the feature; to determine what services are running on a network or a host system. This scan bypasses the traditional firewalls and network filters. This attack helps the adversary identify active services and the hosts' operating system. Looking for connection requests to closed services within in a specific time window can help detect this attack.
Ping sweep: Snooping performed against all operating systems using ICMP where an adversary performs a surveillance sweep to determine which hosts are responding on a network. Information obtained from surveillance is useful to an adversary in launching automated attacks or in making the vulnerable hosts as stepping stones for future distributed attacks. This attack helps the adversary identify active machines on the network and might degrade services for legitimate users. Ping sweep if repeated continuously or launched in a coordinated fashion might result into a low level denial of service attack. Looking for multiple ping requests, destined for all possible machines on a network, all coming from the same host or within a specific time window from multiple hosts can help detect this attack.
UDP scan: Probing attack performed against all operating systems using UDP where an adversary sends 0 byte UDP packets to each UDP service on the target machine to determine which services are running on the victims' machine. This attack helps the adversary identify vulnerable UDP services on victims' network. This information is mostly used to launch automated distributed and coordinated denial of service attacks. Looking for multiple UDP packets with 0 bytes within a specific time window, can help detect this attack.
Null scan: Probing attack performed against all operating systems except Window 95/NT where an adversary turns off all flag options (FIN, URG, PUSH, etc.). This attack helps identify victims' operating system; by sending connection requests to services running on the host machine. Looking for multiple connection requests with all the flags turned off within a specific time window, destined for all possible machines on a network, can help prevent this attack.

Table 4
Probe attacks used in CIA implementation and evaluation

Attack type	Service	Mechanism	Effect of the attack
SYN stealth	Multiple	Abuse of feature	Identifies active machines
FIN stealth	Multiple	Abuse of feature	Identifies active services
Ping sweep	ICMP	Abuse of feature	Identifies active machines
UDP scan	Multiple	Abuse of feature	Identifies active UDP services
Null scan	Multiple	Abuse of feature	Identifies active services
IP scan	Multiple	Abuse of feature	Identifies active protocols
ACK Scan	Multiple	Abuse of feature	Identifies the firewall mechanism (stateful or simple network filter)
Window scan	Multiple	Misconfiguration	Identifies active services
RCP scan	Multiple	Abuse of feature	Identifies active remote procedure call ports (RPC)

IP scan: Snooping performed against all operating systems using raw IP packets without any specified future protocol header. An adversary sends raw IP packets without any specific future protocol header to each specific protocol on the victims' machine. If an ICMP message stating protocol unreachable is received, then it's assumed that specific protocol is not in use. This attack helps identify all the supported protocols on a victims' network. Looking for multiple connection requests without a specific service within a specific time can help detect this attack.

ACK scan: Snooping attack performed to map firewall rule sets where an adversary sends ACK packets (random acknowledgement/sequence numbers) to specific ports. If RST comes back, the specified port is classified as ''unfiltered''. If nothing comes back or an ICMP error message comes, the specified port is classified as ''filtered''. This attack helps identify filtered services and a type of firewall a victims' network has. Looking for random ACK packets can help detect this attack.

Window scan: Probing attack performed against all operating systems using the vulnerability in TCP window size reporting. This attack helps the adversary identify active services as well as filtered services on a victims' machine.

RCP scan: Snooping performed against all operating systems using multiple services to identify active remote procedure call services. This attack helps the adversary identify active remote procedure call services as well as the associated program and version numbers. This information is mostly used to execute arbitrary code by the adversary on a victims' machine. Looking for multiple connection requests to specific remote procedure call services within a specific time, can help detect this attack.

6. CIA system and implementation

Computer probes that are intended to discover information of a computer system can be detected by careful observation of network packets. Probing tools in an effort to identify host information send connection requests to closed ports and non-existing systems. Knowledge of how a network and its hosts are being used will help in distinguishing between normal activity and probes. The primary goal of CIA is to detect probes at the host level. The implemented host agent comprises three components: a data collection module for parsing network packets, a data analysis module for intrusion determination and a response module. The network packet parser uses the WINPCAP library to capture network packets and extracts the relevant features required for classification. The output of the parser includes seven features: (1) duration of the connection to the target machine, (2) protocol used to connect, (3) service type, (4) number of source bytes, (5) number of destination bytes, (6) number of packets sent and (7) number of packets received.

Feature set for our experiments is chosen based on our offline feature-ranking results described in Section 4. Network parser reformats the extracted features to input a classifier to detect probes among other normal network packets. Once the feature set is being constructed it is fed into a suite of classifiers. Classifiers used in our experiments are SVM, MARS and LGP. Output from the classifier is the classification of the connection into normal activity or probe. If a connection is classified as probe a classifier sends a message to the server using TCP/IP sockets and the boundary controllers are updated for necessary response with human intervention to block malicious activity (Fig. 5).

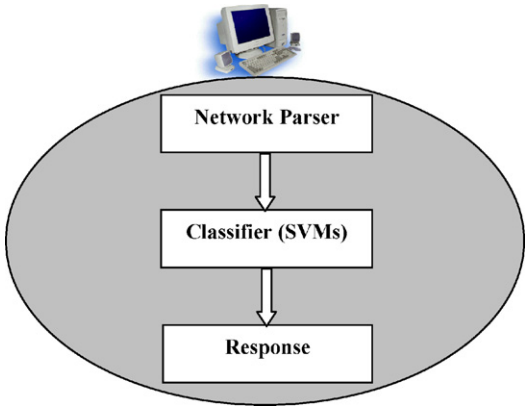

Fig. 5. CIA prototype implementation.

7. Evaluation

Network packets contain information of protocol and service used to establish connection between a client and the server. Network services have an expected number of bytes of data to be passed between a client and the server. If data flow is too little or too much it indicates a suspicion in a connection established that indicates a misuse in service. Using this information normal and probing activities can be separated. In our evaluation, we perform binary classification (normal/probe). The (training and testing) data set contains 10,369 data points generated from normal traffic and probes. The set of 5036 training data and 5333 testing data are divided into two classes: normal and probe. Two separate data sets of sizes 5036 and 5333 are used for training and validating the performance of CIA using SVM, MARS and LGP as classifiers. Tables 5–7

Table 5
Performance of MARS

Class	Normal	Probe	Accuracy (%)
Normal	2018	18	99.12
Probe	0	3297	100.00

Table 6
Performance of SVMs

Class	Normal	Probe	Accuracy (%)
Normal	2031	5	99.75
Probe	1	3296	99.99

Table 7
LGP parameter settings

Parameter	Setting
Population size	512
Tournament size	4
Max no of tournaments	120,000
Mutation frequency	90.94%
Crossover frequency	34.21%
No of demes	10
Max program size	128
Target subset size	100

Table 8
Performance of LGP

Class	Normal	Probe	Accuracy (%)
Normal	2036	0	100.00
Probe	0	3297	100.00

summarize the overall classification accuracy of CIA using MARS, SVM and LGP.

7.1. Experiments using MARS as a classifier

We use five basis functions and selected a setting of minimum observation between knots as 10. The MARS training mode is being set to the lowest level to gain higher accuracy rates. A MARS model is employed to perform binary classification (normal and probe). The objective is to separate normal and probe patterns. Table 5 summarizes the results of MARS as a classifier.

7.2. Experiments using SVMs as a classifier

We used the radial basis function (RBF) kernel function that defines the feature space in which the training set examples will be classified. Table 6 summarizes the results of the experiments using SVMs.

7.3. Experiments using LGPs as a classifier

The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system. The population space has been subdivided into multiple subpopulation or demes. Migration of individuals among the subpopulations causes evolution of the entire population. It helps to maintain diversity in the population, as migration is restricted among the demes. Moreover, the tendency towards a bad local minimum in one deme can be countered by other demes with better search directions. The various LGP search parameters are the mutation frequency, crossover frequency and the reproduction frequency: the crossover operator acts by exchanging sequences of instructions between two tournament winners. Steady state genetic programming approach was used to manage the memory more effectively. After a trial and error approach, the following parameter settings were used for the experiments (Table 8).

8. Conclusions

Computational intelligent agents-based system that is capable of detecting stealthy probes at the host level is being implemented and the results obtained are demonstrated in this paper. A comparison of different computational intelligent techniques is also given. Linear genetic programming technique outperformed SVM and MARS with a 100% detection rate on the test dataset.

The proposed multi-agent framework can operate asynchronously and in parallel, and hence could be useful in monitoring huge networks. By adding new agents, the system can be easily adapted to an increased problem size. Due to the interaction of different agents, failure of one agent may not degrade the overall detection performance of the network. When compared to a centralized system, the proposed distributed framework might be also more cost effective and efficient.

Our future work will be targeted towards multiple agent-based intrusion detection systems that can detect different types of attacks at the host and developing active response mechanisms in an event of intrusion.

Acknowledgments

Support for this research received from Institute for Complex Additive Systems Analysis (ICASA, a division of New Mexico Tech), Department of Defense and NSF IASP capacity building grant is gratefully acknowledged.

References

[1] A.K. Ghosh, Learning program behavior profiles for intrusion detection, in: Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring, 1999.
[2] J. Cannady, Applying neural networks for misuse detection, in: Proceedings of the 21st National Information Systems Security Conference, 1998, pp. 368–381.
[3] J. Ryan, M-J. Lin, R. Miikkulainen, Intrusion detection with neural networks, in: Advances in Neural Information Processing Systems, vol. 10, MIT Press, Cambridge, MA, 1998.
[4] H. Debar, M. Becke, D. Siboni, A neural network component for an intrusion detection system, in: Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, 1992, pp. 240–250.
[5] H. Debar, B. Dorizzi, An application of a recurrent network to an intrusion detection system, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 1992, pp. 78–83.
[6] S. Mukkamala, G. Janoski, A.H. Sung, Intrusion detection using neural networks and support vector machines, in: Proceedings of the IEEE International Joint Conference on Neural Networks, 2002, pp. 1702–1707.
[7] J. Luo, S.M. Bridges, Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection, International Journal of Intelligent Systems, John Wiley & Sons, vol. 15, no. 8, pp. 687–704, 2000.
[8] M. Cramer, et al., New methods of intrusion detection using control-loop measurement, in: Proceedings of the Technology in Information Security Conference (TISC), 1995, pp. 1–10.
[9] J. Stolfo, F. Wei, W. Lee, A. Prodromidis, P.K. Chan, Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection, Results from the JAM Project by Salvatore, 1999.
[10] S. Mukkamala, A.H. Sung, Identifying key features for intrusion detection using neural networks, in: Proceedings of the ICCC International Conference on Computer Communications, 2002, pp. 1132–1138.

[11] S. Mukkamala, A.H. Sung. Feature selection for intrusion detection using neural networks and support vector machines, to appear in Journal of the Transportation Research Board (of the National Academies), 2003.
[12] S. Mukkamala, A.H. Sung, Identifying significant features for network forensic analysis using artificial intelligence techniques, International Journal on Digital Evidence, IJDE vol. 1, issue 4, 2003.
[13] D. Denning, An intrusion-detection model, IEEE Trans. Software Eng. SE-13 (2) (1987) 222–232.
[14] S. Kumar, E.H. Spafford, An application of pattern matching in intrusion detection, Technical Report CSD-TR-94-013, Purdue University, 1994.
[15] S. Staniford, J. Hoagland, J. McAlerney, Practical automated detection of stealthy port scans, J. Computer Security 10 (1/2) (2002) 105–136.
[16] R. Basu, K.R. Cunningham, S.E. Webster, P.R. Lippmann, Detecting low-profile probes and novel denial of service attacks, in: Proceedings of the 2001 IEEE Workshop on Information Assurance, US Military Academy, (2001.
[17] S. Mukkamala, A.H. Sung, A. Abraham, Distributed multi-intelligent agent framework for detection of stealthy probes, in: Third International Conference on Hybrid Intelligent Systems, Design and Application of Hybrid Intelligent Systems, IOS Press, Amsterdam, The Netherlands, (2003), pp. 116–125.
[18] J.H. Friedman, Multivariate adaptive regression splines, Anal. Stat. 19 (1991) 1–141.
[19] D. Steinberg, P.L. Colla, M. Kerry, MARS User Guide, Salford Systems, San Diego, CA, 1999.
[20] A. Abraham, D. Steinberg, MARS: still an alien planet in soft computing? In: Vassil N. Alexandrov, et al. (Eds.), Lecture Notes in Computer Science 2074, Springer–Verlag Germany, USA, pp. 235–244, 2001.
[21] T. Joachims, Making large-scale SVM learning practical, LS8-Report, University of Dortmund, LS VIII-Report, 1998.
[22] T. Joachims, SVMlight is an implementation of support vector machines (SVMs) in C, http://ais.gmd.de/∼thorsten/svm_light, University of Dortmund, Collaborative Research Center on Complexity Reduction in Multivariate Data (SFB475), 2000.
[23] V.N. Vladimir, The Nature of Statistical Learning Theory, Springer, 1995.
[24] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann Publishers, Inc., 1998.
[25] K. Kendall, A database of computer attacks for the evaluation of intrusion detection systems, Master's Thesis, Massachusetts Institute of Technology, 1998.
[26] S.E. Webster, The development and analysis of intrusion detection algorithms, S.M. Thesis, Massachusetts Institute of Technology, 1998.
[27] K. Krawiec, Genetic programming-based construction of features for machine learning and knowledge discovery tasks, Genet. Program. Evolv. Machines 3 (4) (2002) 329–343.
[28] Network mapper, http://www.insecure.org/nmap/.