A Security-policy Driven Distributed Incident Response Generator

Bel G Raggad[1]
Emilio Collar Jr.[2]
Abhi Pandya[3]

[1]Pace University, New York, USA
[2] Western Connecticut State University, Danbury, USA
[3]Florida Atlantic University, Boca Raton, USA

Abstract. While the information security literature reported great advances in Intrusion Detection Systems (IDS) capabilities, it those systems have neglected their weaknesses in dealing with events that are detected when some state variable values are out of range but declared unknown because they could not find their definitions in the IDS databases. A Distributed Incident Response Generator (DIRG) is simply a distributed decision support system designed to generate incident responses in a distributed computing environment when the existing IDS system suspects an event that does not correspond to a known intrusion, residing in its databases. Since the suspected event is unknown to the IDS, the security administrator would have a great deal of uncertainty that should be feasibly managed to plan the appropriate incident response actions in a timely manner. In addition to the uncertainty associated with the suspected event, many data and information assets may be remotely located in a large organization, and an intrusion may be detected first in one location but not in others. In this case, a distributed stateful inspection of critical resources can help identify security incidents early enough to prevent further security compromises in the distributed computing environment. Every time the security administrator suspects an intrusion based on an IDS message of an unknown event, he/she creates several scenarios of possible security incidents that are compatible with a multiple-domain security knowledge designed to enforce the security policy of the organization. Values of state variables are collected from remote locations through distributed stateful inspection activities for the purpose of obtaining enough evidence to plan incident responses for the unknown event. The type of data involved in intrusion detection when ample uncertainty is present is often not suitable to formal statistical models and Bayesian modeling is not appropriate. This article proposes the adoption of Dempster and Shafer theory to process the intrusion data for the unknown event. The DIRG system engine transforms intrusion data into a belief structure using (1) the possible incident scenarios, (2) the consolidated stateful inspection data obtained throughout the distributed computing environment, and (3) the feasible security knowledge associated with the enforcement of the organization's security policy. Belief values associated with various incident scenarios are then derived and evaluated to choose the most appropriate scenario for which an automatic incident response is generated. This article also provides a numerical example demonstrating the working of the DIRG system.

Key words: intrusion detection, incident response, distributed computing, decision support system, security risk, stateful inspection.

Introduction

The literature went many different ways in surveying intrusion detection system taxonomies. Each reported taxonomy may serve one investigation purpose but not others depending on the objective of the study. In this article, we adopt Lazarevic, et al.'s

intrusion taxonomy because it provides information about several features that we need in studying unknown intrusions picked by a working IDS system (Lazarevic et al., 2005). The IDS system, in this case, has sensors that detect out of range state variables to indicate a suspected intrusion without knowing the exact identity of the intrusion being executed. Lazarevic, et al. proposed an inclusive taxonomy based on attack type, number of network connections involved in the attack, source of the attack, computing environment, and automation level (Lazarevic et al., 2005; Sommestad and Hunstad, 2013).

Attack types have been organized by most of the literature as DoS, probing attacks, compromising attacks, and viruses. Other classifications of attacks will most often lead to one of those types (Kendall, 1998).

In addition to denying the use of resources and services to authorized users, DoS attacks aim at diminishing or fully eliminating the availability of computing resources by stalling networks, computers, or programs (Marchette, 2001). There are operating systems attacks that exploit code bugs and flaws, as well as network attacks that exploit vulnerabilities in communication protocols. There are also distributed DoS (DDoS) attacks where multiple machines are deployed to eliminate availability of computer resources (Mirkovic and Reiher, 2004: 39-54; Peng, 2002). Early detection of DDoS attacks is very important so the attacks can be studied, especially the progression of attacking activities, so that quick responses can be planned for the purpose of limiting the ongoing effects. The probing attacks start by conducting surveillance and scanning to identify feasible victims. The scanning is needed to find the IP addresses they can exploit and assemble all the information they need about the victims' operating systems and offered services. Once enough vulnerabilities are known, attackers can then plan their attacks to inflict harm and discover more vulnerabilities (Ertoz et al., 2004; Jung et al., 2004; Robertson et al., 2003; Staniford et al., 2002: 105-136). The compromising attacks can be performed by insiders or outsiders. These attacks aim at full penetration of systems to gain privileged access to computing resources and compromise their security. Outsiders who are not legitimate users of the system can gain access as users, or access the root directory, and break into systems. The higher the privilege obtained the more harm is inflicted to the victim's system.

On the other hand, insiders often have legitimate accounts with given privileges. However, they can intentionally misuse their authorized services and elevate their privileges by exploiting discovered vulnerabilities in the systems where they are legitimate users (Feng et al., 2003). Viruses, with all their categories, consist of programs that replicate on computer systems and propagate through networks. They can erase files on the hard disk and install malicious programs. Categories of viruses have been defined in terms of their environments, their operating systems, the code they use, and their destructive power (Feng et al., 2003). The number of network connections involved in an attack is also a very important feature of attacks. For example, DoS, probing, and worms are known to use multiple network connections. Conversely, buffer overflow attacks typically use single network connections. Most compromising attacks are often very focused attacks that lead to penetration and, hence, use single network connections across a small number of networks.

The source of attack is also an important indication of the distribution of the attacks. Planning responses to attacks is more effective when the origin of the attack is known. The environment of the attack is also a mandatory feature we need to know. Planning incident responses requires knowledge of the computing environment where the attack

is taking place. Wireless attacks will require different preventive and corrective actions compared to connected environments. In a connected environment, it is also consequential to know the type of environment the attack is taking place, whether the attack is hitting a firewall, host machine, an entire network, or a VPN environment, for example.

The automation level will indicate whether we are dealing with a manual attack, an automatic process, or both. Manual scanning of systems is not as fast and as propagating as automatic scanning and should be handled differently (Staniford et al., 2002: 105-136).

An IDS

As in NIST (The National Institute of Standards and Technology), we view intrusion detection to be the monitoring of a computing environment and its inspection of signs of intrusion and attempts to compromise the confidentiality, integrity, and availability of its computing resources (Bace and Mell, 2001).

Intrusions can be performed by either hacker when they access a computing environment or by legitimate users who abuse their privileges. An intrusion detection system is often made of both software and hardware tools designed to monitor a computing environment. Although Dorothy Denning's work in designing the first intrusion detection system (Denning, 1987: 222-232) is dated, intrusion detection systems continue to be designed the same way. That is, intrusion detection systems are designed to 1) gather data through consolidating signals from sensors plugged around the network, 2) detect intrusive activities based on the sensors' information, 3) sequentially populate databases given sensors' information, and 4) configure tools for defining the current state of the system so that effective responses can be planned in a timely manner.

The literature advanced several features that need to be included in an intrusion detection system (Debar et al., 1999; Porras and Valdes, 1998): Prediction Performance, Time Performance, and Fault Tolerance.

Prediction accuracy alone is insufficient for modeling IDS performance. For instance, a valid IDS prediction accuracy value may classify network traffic as safe because > 98% of the traffic is legitimate data. However, the 98% does not make sense due to the < 2% of network traffic considered suspicious. If this performance criterion is sound, then most IDS systems will be sound and most transmitted data will be safe since IDS accuracy has exceeded 98%. That is, the performance of an IDS system should be linked to the quality of its output in terms of its detection rate and false alarms (Tavallaee at al., 2010: 516-524; Wang and Liu, 2008).

An IDS should correctly detect intrusions and those detected intrusions can only be real intrusions and not legitimate activities. Two performance measures are critical to measure the effectiveness of an IDS system: the detection rate and the false positive rate. The detection rate is computed as the number of correctly detected intrusions divided by the total number of intrusions. The false alarm rate is computed as the number of legitimate activities that are reported as intrusions divided by the total number of intrusions (Porras and Valdes, 1998).

The time performance of an IDS measures the time lapse before an intrusion is detected including the processing time and the intrusion transmission time. The shorter the time the earlier a security administrator could plan a response (Jamdagni et al., 2013: 811-824).

The Distributed Incident Response Generator (DIRG)

This study assumes a working IDS system that can detect all known intrusions, as well as alerting the security administrator to plan the appropriate incident response. Occasionally, the working IDS system suspects some events that cannot be classified as known intrusions because the adopted security attack is not defined in the system's intrusion knowledge base. In those cases, the IDS system alerts the security administrator of the suspected event and triggers the DIRG to provide assistance in recognizing the unknown intrusion and to later generate recommendations of the most appropriate actions that should be taken.

As shown in Fig. 1, when the DIRG is triggered as discussed earlier, the security administrator has to start the required input preparation for the DIRG: an event recognition record, distributed data containing among other things the current values of state variables, and the feasible specific security domain relevant to the event in question.

At this point of the decision process, the event conditions only constitute an initial guess by the security administrator on the possible scenario(s) given the IDS report describing an unknown event. Information about the initial event conditions is then defined by the security administrator and needs to be evaluated using values of state variables taken at the organization's distributed remote locations. Real-time distributed processing is essential to prevent the propagation of the suspected event.

The security knowledge base contains, for each security domain, the relevant security knowledge associated with the suspected event conditions, their state variables, intrusion properties, and confirmed corrective actions, in addition to all established certainty factors, if any. The knowledge availability is assured by the security policy and the working IDS system where the DIRG is installed. In general, however, there are a variety of knowledge discovery and engineering techniques that can produce security knowledge needed for the DIRG system. Classification techniques, such as clustering, neural networks, decision trees, nearest neighbors, and pattern recognition are all examples of knowledge discovery tools that can be used to create knowledge or to enhance the existing knowledge bases. The security knowledge base may also be created by training, which uses validated knowledge patterns collected throughout the life cycle of the organization's security management system (Modi et al., 2013: 42-57).

The DIRG system can also be activated automatically. This activation, however, requires the addition of a module capable of identifying those event conditions associated with the information received from the working IDS system and any scenarios assembled by the security administrator in the event record. The security administrator can then connect to the DIRG system and enter his/her incident scenarios to complete the security administrator's part of the event record. This feature of the system, however, is considered beyond the scope of this article and may be studied in a future version of this article.

The proposed DIRG system can be integrated in any computing environment with a working IDS system without affecting the operation of existing components. The quality of security decision support information generated by the DIRG system will depend on 1) the quality of data and knowledge received from remote locations, 2) the quality of the IDS system's information on the unknown intrusion, and 3) the quality of a specific knowledge base relevant to the event in question. If the distributed data on state variables or the knowledge bases contain errors or if the event record is not well defined, then the DIRG will not be aware of those errors and the resulting erroneous information will be processed as valid information and the output of the inference engine will certainly be affected. Eliminating data inconsistencies in the databases, validating the contents of the

knowledge bases, or detecting any other erroneous information in the DIRG input are beyond the scope of this article and may be studied in a future research project.

Most critical distributed information resources have their own security policy that defines its acceptable behavior as defined by its owners. Each behavior rule specifies all state variables that are indicative of an information resource's acceptable behavior. Each state is defined either as a category or as a number. The categorical values are specified to indicate certain acceptable behaviors of the information resources. If the current state for one resource does not belong to the set of acceptable states, then there is a problem. Hence, this is an undesired event.

Sometimes the undesired states are categorized as an acceptable state. If the current, acceptable state belongs to the set of undesired states then there is a problem; that is, we have an undesired event. For example, a promiscuous workstation in a local area network is an undesired state. This workstation compromises the confidentiality of data exchanged on the network for the rest of the users. The state variables may also be specified as ranges between a minimum and a maximum number, such as the bandwidth consumption state variable. If a current state variable for an information resource is out of range, then there is a problem; hence, this is an undesired event.

As depicted in Fig. 1, the inference engine process is organized into the following steps:

Step 1: The working IDS system detects an event that does not correspond to a known intrusion. This information is stored in the event recognition record.

Step 2: The security administrator studies the situation and identifies incident scenarios that can be responded to. The information of 1) and 2) is stored in the event record.

Step 3: Based on the event record, the security administrator selects all security knowledge relevant to the suspected event. This is called the feasible specific domain knowledge.

Step 4: Extract from the distributed remote location the values of the state variables and any relevant information needed in the recognition of the suspected event.

Step 5: The DIRG system now has all needed input ready: event record, remote data on state variables, and feasible knowledge relevant to the suspected event conditions defined by the security administrator. This input is then submitted to the inference engine.

Step 6: The inference engine processes the input and generates recommendations to the security administrator.

Step 7: The security administrator interprets the system's recommendations and plans the most appropriate incident responses.

Step 8: The security administrator accepts the DIRG recommendations. This ends the planning of an incident response.

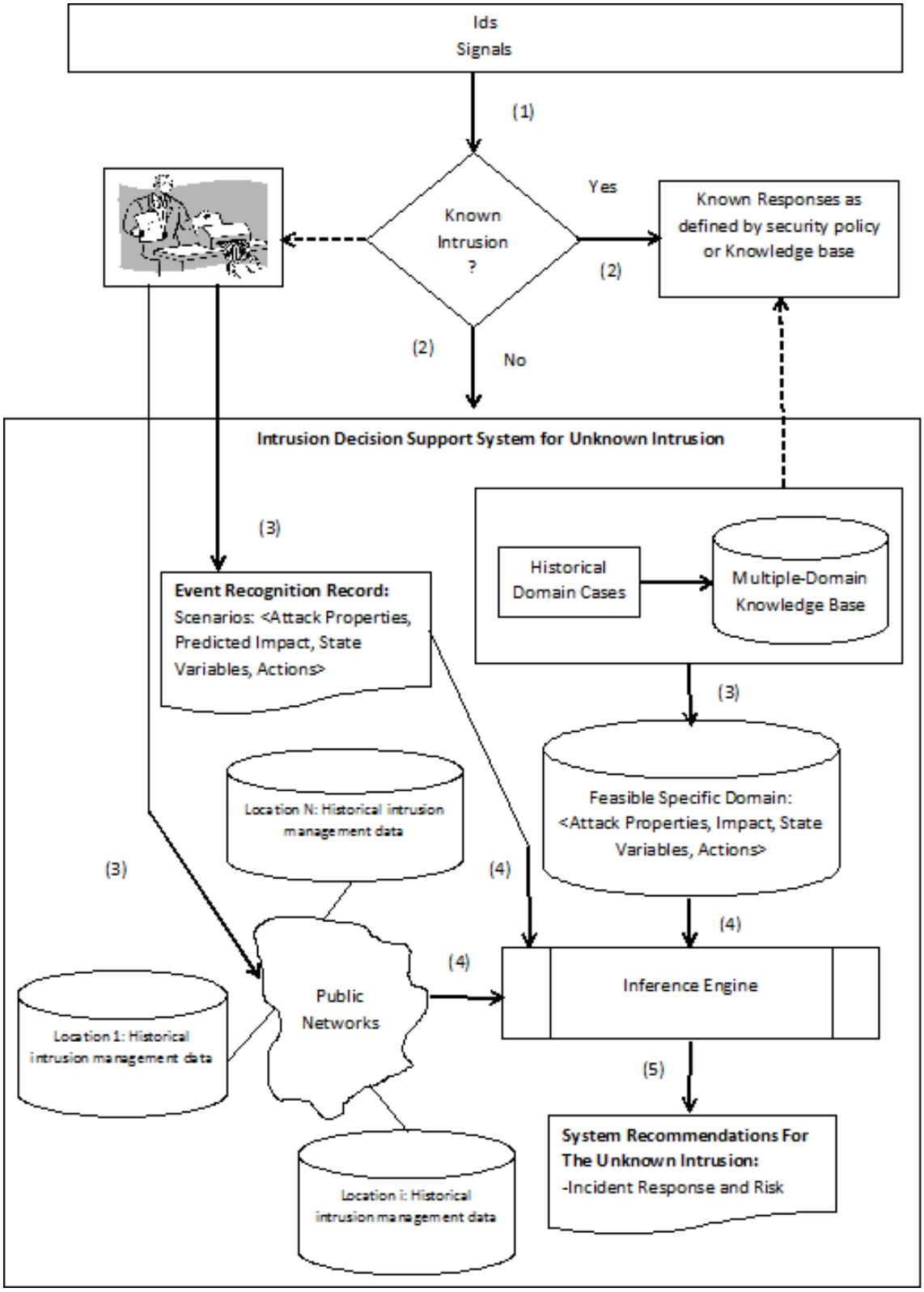

Fig. 1. Working of the Distributed Incident Response Generator (DIRG)

Let us assume that our system architecture and connectivity are adequately configured so our DIRG can be implemented. The DIRG is scalable as we can add as

many historical intrusion management databases and feasible security knowledge bases as necessary. This architecture is an independent addition and cannot affect any existing technology including the IDS databases and knowledge bases. This system requires the presence of a security administrator who needs to receive IDS messages of the detected, but unknown intrusions in real-time. The security administrator provides all the information in the event recognition record, which is later entered into the DIRG to generate decision support information. This information is used for defining new intrusions and planning an incident response for those new intrusions.

Dempster and Shafer Theory

Dempster and Shafer theory considers a frame of discernment Ω where all relevant objects reside. A measure from 2Ω, to [0 1] is called a basic probability assignment and is defined on subsets of Ω, to model the uncertainty associated with the propositions of interest. A proposition is simply a subset of the frame of discernment for which a basic probability assignment is needed. Mass values m can be assigned to propositions to represent the uncertainty associated with them as follows:

$$m: 2^{\Omega} \rightarrow [0\ 1]$$

$$m(\emptyset) = 0$$
$$\sum_{X \subseteq U} m(X) = 1.$$

The mass m(X) represents the belief exactly committed to X, which is the exact evidence that the value of Ω is in X. Most often, whenever we have m(X) > 0, then this means that there is real evidence that a value of Ω would be in X. We define X as a *focal element*.

Then, given all the evidence in hand, made of all the focal elements and their mass values, we can compute the total belief provided by available evidence for a proposition X, as follows:

$$Bel(X) = \sum_{Y \subseteq X} m(Y).$$

The belief value Bel(X) is the total belief committed to X, that is, the mass of X itself plus the mass values attached to all subsets of X. The value Bel(X) is then the total positive effect the evidence has on the value of Ω being in A.

In addition to the belief value Bel(X) there is another quantity Pl(X) called the *plausibility* of X that expresses the remaining uncertainty associated with the negation of the proposition X. The plausibility function is defined as follows:

$$Pl: 2^{\Omega} \rightarrow [0\ 1]$$
$$Pl(X) = \sum_{X \cap Y \neq \emptyset} m(Y).$$

The value Pl(X) is the sum of mass of X and mass values of all subsets that intersect with X. The plausibility of X measures the extent to which the available evidence fails to negate X, and should hence be equal to 1-Bel (Not X).

The inference mechanism

Before proceeding further in our discussion of the inference engine, we need to

define some variables as follows:

V= $\prod_{j=1,MV} V_j$ = Structured security domain space;

V_j = Attribute number j in the security domain space, made of countable objects in Dom (V_j), the domain of the attribute;

F = a subset of V containing the feasible security space where the solutions are relevant to the event record;

e = event compatible with the structure of the security domain space V;

Δ = Partial order relation on all data sets;

R(a) = Risk associated with taking the assertion a.

There are three input types to the inference engine: 1) the event record, 2) the distributed data from remote locations, and 3) the feasible security knowledge corresponding to the suspected event conditions as defined by the security administrator given the detection of an undesired event. Our DIRG system assumes a common format made of hyper-tables for the event record, the intrusion management data, and the feasible security knowledge bases.

A table is a set of data arranged in rows and columns. Most often, the rows are called tuples and the columns are called attributes. The i^{th} value t_i in a tuple corresponds to the i^{th} attribute A_i of the table and belongs to the domain V_i of the i^{th} attribute. A tuple t is denoted $t = (t_1, \ldots, t_N)$ where $t_i \varepsilon V_i$, I = 1, N.

While a row in a table is a tuple of singled values of the attributes, a hypertuple is instead a tuple of subsets of the attributes. That is, a hypertuple τ is denoted $\tau = < \tau_1, \ldots, \tau_N >$ where τ_i is a subset of V_i, I = 1, N. A hypertable is simply a table of hypertuples and hyperdata is simply data made of hypertables.

Consider then an event record e, $e=\{e_1, \ldots, e_{Me}\}$ where $e^k = (e^k_1, \ldots, e^k_{Ne})$, $k = \{1_e, M\}$ and where $\{e^k_1, \ldots, e^k_{Me}\}$ is a subset of $\{V_1, \ldots, V_{Mv}\}$. Also let Δ be a partial order relation on all the data sets on hand. If x and y are elements of a set E, we say that $x \Delta y$ if and only if $x \subseteq y$. The inclusion defines the amount of support x provides to y, or alternatively, the amount of compatibility between x and y.

Given two subsets E and G and x in G, we define the evidence support $S_G(x)$ of x in G as the set of y in G such that $y \Delta x$. That is, $S_G(x) = \{y \varepsilon G$, such that $y \Delta x\}$. The subset G is a posit with respect to the partial order relation Δ and it may hence have elements that are related to x (fully compatible) and others that are not related to x (not fully compatible). Only the compatible elements y in G such that $y \Delta x$ are accepted to support x.

How to derive an intrusion belief structure?

As discussed above, there are three input types to the inference engine: 1) the event recognition, 2) the distributed data describing state variables from remote locations, and 3) the feasible security knowledge corresponding to the security domain associated with the current suspected event.

The intrusion belief structure is the ultimate output of the inference mechanism because it produces a consolidation of the three types of evidence needed to diagnose the suspected event conditions, which is required to plan the appropriate incident response. As shown in Fig. 2, the basic probability assignment m(e) attributed to a candidate event scenario proposition e^i is computed as the cardinal of the support $S_D(e^i)$ that D gives to e^i divided by the support $S_D(F)$ that D gives to the feasible security

knowledge set F. Since $S_D(F)$ is a normalization factor, then this function m(.) from 2^F to [0 1] is obviously a mass function. Other properties of this function are described in more details in Wang and McClean (2008: 455-465) and Xu et al., 2011: 385-399.

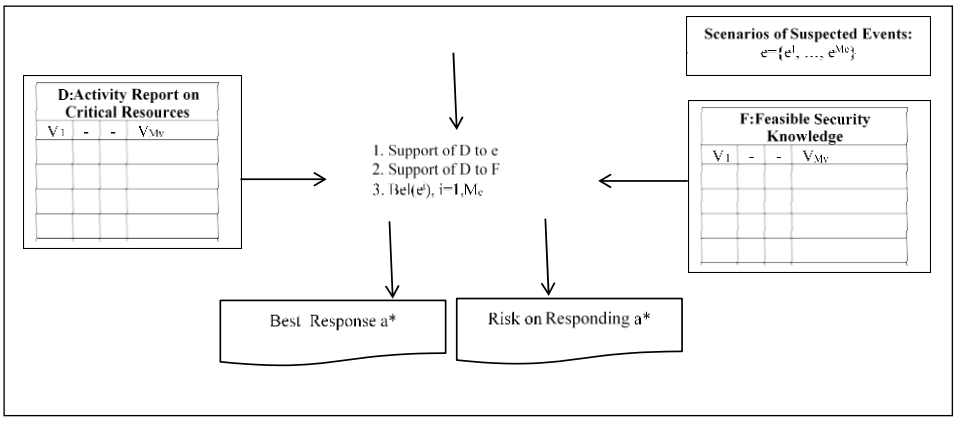

Fig. 2. Deriving a belief structure for the current intrusion

The security administrator computation process consists of the steps discussed above. Those steps are performed in accordance with a structured algorithm that could be easily translated into computer code. This algorithm is designed as follows:

Algorithm: Find the best security administrator's assertion a* given an event recognition record:

Begin

1. Consider an event recognition record $e = \{e^1, \ldots, e^{Me}\}$ where $e^k = (e^k_1, \ldots, e^k_{Mv})$, $k = \{1, M_e\}$ and where

$\{e^k_1, \ldots, e^k_{Mv}\}$ is in $V_1 x \ldots x V_{Mv}$.

2. For i=1, Me, do the following:

a. Compute $S_G(e_i) = |e_i \Delta G|$;

b. Compute $S_G(F) = |F \Delta G|$;

c. Compute $m_G(e_i) = |e_i \Delta G| / |F \Delta G|$;

d. Compute $Bel(e_i)$.

e. Compute $e^* = \text{argmax}_{[ei \varepsilon e]} Bel(e_i)$

f. Compute $a^* = \text{Projection}_{[Assertions]} (e^*)$

g. Compute R(a*)

End.

We next explain the working of the inference mechanism algorithm which guides the security administrator to process available evidence and produce the belief structure from which the belief values are induced. At this point, we obtained a belief structure, as follows:

$$m_D: 2^V \rightarrow [0\ 1]$$

$$m_D(x) = |S_G(x)| / |S_G(F)|$$

where $S_G(F) = \{\{y \varepsilon G \text{ such that } y \Delta x\}, x \varepsilon F\}$

Given all the evidence stored in the belief structure just obtained, we can compute the total belief provided by the available evidence for a proposition e, as follows:

$$Bel(e) = \sum_{Y \subseteq e} m(Y)$$

The belief value Bel(e) is the total belief committed to e, that is, the mass of e itself plus the mass values attached to all subsets of e. The value Bel(e) is then the total positive effect the evidence has on the value of Ω being in e.

At this point, we have the information we need to evaluate the assertions on event conditions identified in the event recognition record by the security administrator. We need to compute the belief values of all assertions expressed in the candidate propositions $\{e^i, i = 1, Me\}$. Once the best proposition e* is produced we need to project over the assertion attribute to obtain the best security administrator assertion a*. We then have the following:

$$e^* = \text{argmax}_{[ei \varepsilon e]} Bel(e^i)$$
$$a^* = \text{Projection}_{[\text{Assertions}]} (e^*)$$

The intrusion analysis ends with the security administrator planning an incident response according to the DIRG recommendations.

Finally, as in any decision process under uncertainty, there will be always risk associated with the security administration decision process. This risk R(a*) is defined as the plausibility of the evidence against the selected assertion; that is, the plausibility of the negation of e*. This amount is also equal 1 minus the belief of e*. We then have:

$$R(a^*) = Pl(\text{not } a^*) = 1 - Bel(e^*)$$

Unfortunately, in practice, the computations above may be very lengthy and expensive when the feasible security space F is large. Also, the normalization factor can be very costly when F is large. The normalization factor is originally computed as the total support of D in F. That is, we have to consider every element in F and count the number of elements in D that are compatible with this element in F. This process is too long and too expensive. We can show, as in Wang and Liu (2008), Wang and McClean (2008: 455-465), Xu et al. (2011: 385-399), that we can instead consider the elements in D and count the number of elements in F that are compatible with this element in F. This may be written as $\sum_{f \varepsilon F} [\{d \varepsilon D \text{ such that } d \Delta f\}] = \sum_{d \varepsilon D} [\{f \varepsilon F \text{ such that } d \Delta f\}]$.

Then, as in Wang and Liu (2008), Wang and McClean (2008: 455-465), we can compute the mass values as follows:

$$S_D(e) = \prod_{i=1, Me} 2^{|ai|-|ei|};$$
$$S_D(F) = \sum_{x \varepsilon F} \{\prod i=1, Me\ 2^{|ai|-|xi|}\};$$
$$m_D(e) = |S_D(e)| / |S_D(F)|.$$

Numerical example

Consider a working IDS that generated a signal indicating the presence of an

intrusion of unknown type. After the security administrator received the IDS message, he/she proceeded by defining the event recognition record where propositions listing different scenarios of the intrusion situation have been previously defined.

Assume that the event recognition record e is made of two propositions: $e = \{e^1, e^2\}$. These two scenarios are included in the event recognition record provided in Table 1.

$e = \{e^1, e^2\}$

$e^1 = <$ {T3, T2}, {a2, a5}, {L, H}, {L, A}, {L}, {L, H}, {L, H}, {1H}, {U}, {L, H}, {HL} >

$e^2 = <$ {T5}, {a1, a2}, {L, A}, {L, A, H}, {A}, {L, H, {MH}, {U}, {L, A}, {HL} >

Table 1. Event Recognition Record

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
{T3, T2}	{a2, a5}	{L, H}	{L, A}	{L}	{L, H}	{1H}	{U}	{L, H}	{HL}
{T5}	{a1, a2}	{L, A}	{L, A, H}	{A}	{L, H}	{MH}	{U}	{L, A}	{HL}

Assume that the distributed state variable data in remote locations has been consolidated and produced the data stored in Table 2. Also assume that the feasible knowledge available for this type of event is given in Table 3. Incident responses are the actions to take whenever a suspected intrusion is confirmed. The following incident responses are considered:

a1: Structured exception handling using try/catch blocks
a2: Catch and wrap exceptions
a3: Implement a global exception handler
a4: Do not log private data
a5: Use proven platform-provided cryptography
a6: Reduce session timeouts
a7: Secure critical channels
a8: Constrain, reject, and sanitize input.

The types of intrusion recognized by the IDS system are listed as follows:

T1: Elevation of privilege
T2: Information corruption
T3: Information disclosure
T4: Forgery
T5: Denial of Service (DoS)
T6: Scripting

Table 2. Historical Intrusion Management Data

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}
{T3}	{a5}	{L}	{A}	{L}	{H}	{1H}	{U}	{H}	{HL}
{T5}	{a2}	{L}	{L, A, H}	{A}	{L}	{MH}	{U}	{L}	{HL}
{T1}	{a1}	{L}	{H}	{H}	{L}	{1N}	{W}	{H}	{HL}
{T3}	{a5}	{L}	{L}	{L}	{H}	{1H}	{U}	{H}	{HL}
{T5}	{a5}	{A}	{H}	{A}	{H}	{1H}	{W}	{A}	{ML}
{T3}	{a6}	{A}	{H}	{A}	{A}	{MN}	{WL}	{H}	{VHL}
{T5}	{a5}	{L}	{A}	{A}	{A}	{I}	{U}	{H}	{SL}
{T1}	{a1}	{L}	{L}	{A}	{L}	{1H}	{M}	{A}	{HL}

Table 3. Feasible Security Knowledge

V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}

{T3, T2}	{a2, a5}	{L, H}	{L, A}	{L}	{L, H}	{1H}	{U}	{L, H}	{HL}
{T5}	{a1, a2}	{L, A}	{A, H}	{A}	{L, H}	{MH}	{U}	{L, A}	{HL}
{T1, T5}	{a3, a7}	{L, H}	{L, H}	{H}	{L, H}	{1N}	{W}	{L, H}	{HL}
{T3, T4, T6}	{a2, a4, a8}	{L}	{A}	{H}	{L}	{I}	{U}	{L}	{SL}
{T3, T5}	{a5, a7}	{A, H}	{A, H}	{L, H}	{A, H}	{1H}	{W}	{A, H}	{ML}
{T5, T6}	{a6, a8}	{A}	{H}	{A}	{L}	{MN}	{WL}	{A}	{VHL}
{T5, T6}	{a2, a6}	{L, H}	{L, H}	{L, A}	{L, A}	{I, MH}	{U}	{L, H}	{ML, HL}
{T1, T3}	{a1, a5, a6}	{L, A, H}	{L, A, H}	{L, A}	{L, H}	{1H}	{U,M}	{L, A, H}	{HL}

Table 4. Computation of the normalization factor | sD^{α} (F) |

Feasible tuples	$\| sD^{\alpha}$ (x), x in F |
F_1	2
F_2	1
F_3	0
F_4	0
F_5	1
F_6	0
F_7	1
F_8	2
Total:	7

We then obtain the following belief values:

$$m_D(e^1) = |S_D(e^1)| / |S_D(F)| = 2/7 = 0.29$$
$$m_D(e^2) = |S_D(e^2)| / |S_D(F)| = 1/7 = 0.14$$
$$Bel_D(e^1) = .29$$
$$Bel_D(e^2) = .14$$

Risks associated with the security administrator's decision are as follows:

$$R(assertion(e^1) = 1- BelD(e^1) = .71$$
$$R(assertion(e^2) = 1- BelD(e^2) = .86$$

One can then see that the optimal solution for the security administrator is to select the following assertion:

$\{a_1, a_5\}$ = {Catch and wrap exceptions, use proven platform-provided cryptography} that has the highest belief value.

Managerial implications

The DIRG system we proposed fits in any computing environment with an IDS system to enhance the evidence management process that is needed to assure the security of the organization. Without this addition, there will be many incidents that cannot be managed based only on the signals generated by the existing IDS system. Due to the absence of statistically sound models and the existence of intrusion data that does not satisfy the statistical assumptions required by those models, IDS information may be erroneously combined. Consequently, inadequate analytical models may be adopted to

manage this type of evidence, which will lead to imprecise recommendations.

We discussed in this article that because of the structure and incompleteness of intrusion data a Bayesian model was not possible. Also, any other analytical model will be ignoring the feasible security knowledge used by our DIRG system to validate the security administrator's definition of the incident recognition record. The proposed system uses every information capability available in the computing environment to generate sound incident response decision support for the security administrator. We included incident response information from the incident recognition record, historical incident management information, and any available feasible security knowledge to validate the security administrator's decisions.

From the technical side of the proposed system, we proposed a method to collect, assemble, and combine evidence before decision support is generated. Event analysis can benefit from our proposed method and the intrusion management data combined with available relevant security knowledge before producing incident response recommendations. We showed how to transform intrusion data into belief structures that can be combined and processed. Belief measures may be obtained and the incident response assertion that corresponds to the highest belief is the one to be retained. Incident response recommendations are designed according to the retained belief values.

The belief model we presented is not only useful to process decision support in intrusion management, but it is also valuable to any other decision situation where there is decision support under uncertainty and available data in the computing environment. When a Bayesian model cannot be constructed, and only partial information is available, the construction of a belief model is consequential. In this case, optimal decisions can be made and risks assessed in a statistically sound way.

Conclusion

While most of the literature reported great advances in IDS capabilities, it has at the same time neglected the weakness of IDS systems in dealing with events that are detected when some state variable values are out of range but remained unknown because they could not find their definitions in IDS databases. That is, even when sensors detected those undesired events, the IDS systems still failed to identify the corresponding intrusion in its IDS data bases.

This paper discussed how uncertainty is processed and proposed a distributed incident response decision support system that can be added to any IDS environment.

In designing the proposed DIRG system, this paper proposed a method to construct belief structures based on event recognition records and the feasible security space associated with the event recognition process. Security administrator's assertions are processed and optimal incident responses are generated in a risk-driven manner. The paper also provided a numerical example to demonstrate the working of the proposed method.

There is a great deal of uncertainty in many areas in intrusion detection for which the Bayesian formalism is not valid due to the statistical assumptions that cannot be verified in available data. Professionals working at the decision support level in security management can transform available data into belief structures that can be easily combined using Dempster's rule and processed to provide the decision support they need with better accuracy.

Future research paths can investigate how the proposed DIRG system can benefit from adding partial compatibility (instead of full compatibility studied in this article) by

stretching the partial order relation used in processing the evidence support mechanism. Instead of counting as compatible only those subsets of the historical intrusion management data that fully belong to instances of the feasible security knowledge, we can study what could happen if we also count those subsets that only intersect (instead of one being fully included in the other) with hypertuples in the feasible knowledge base. Those intersections, no matter how small they are, hold partial information. The effects of that partial information on the incident response recommendations need to be studied. This concept of partial compatibility may be useful in cases where IDS messages and historical intrusion data are too general to lead to precise incident response recommendations or when we are dealing with intrusions that are new to the feasible security knowledge base.

References

Bace R., Mell, P. (2001). NIST Special Publication on Intrusion Detection Systems. Computer Security Resource Center. Available at: https://csrc.nist.gov/publications/detail/sp/800-31/archive/2001-11-01

Debar, H., M. Dacier, Wespi, A. (1999). Towards a Taxonomy of Intrusion Detection Systems, Computer Networks, 31(8), 805-822. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.626.4329&rep=rep1&type=pdf

Denning, D. (1987). An Intrusion-Detection Model. IEEE Transactions on Software Engineering, 13(2), 222-232. http://dx.doi.org/10.1109/TSE.1987.232894

Ertoz, L., Eilertson, E., Dokas, P., Kumar, V., Long, K. (2004). Detection – Revisited. Army High Performance Computing Research Center Technical Report. Available at: https://www.researchgate.net/profile/Vipin_Kumar26/publication/265810891_Scan_Detection_-Revisited/links/54acb3810cf23c69a2b8391c.pdf

Feng, H., Kolesnikov, O.M., Fogla, P., Lee, W., Gong, W. (2003). Anomaly Detection Using Call Stack Information, In Proceedings of the IEEE Symposium Security and Privacy, Oakland. Available at: https://dl.acm.org/doi/10.5555/829515.830554

Jamdagni A., Tan, Zh., He, X., Nanda, P., Liu, R. (2013). RePIDS: A Multi tier Real-time Payload-based Intrusion Detection System. Computer Networks, 57(3), 811-824. http://dx.doi.org/10.1016/j.comnet.2012.10.002

Jung, J., Paxson, V., Berger, A., Balakrishnan, H. (2004). Fast Portscan Detection Using Sequential Hypothesis Testing. Proceedings of the IEEE Symposium on Security and Privacy, Oakland. http://dx.doi.org/10.1109/SECPRI.2004.1301325

Kendall, K. (1998). A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems, Massachusetts Institute of Technology Master's Thesis.

Lazarevic, A., Kumar, V., Srivastava, J. (2005). Intrusion Detection: A Survey. In: Kumar V., Srivastava J., Lazarevic A. (Eds.). Managing Cyber Threats. Massive Computing, Vol. 5. Boston: Springer. Available at: https://pdfs.semanticscholar.org/e241/3f14a014603253815398e56c7fee0ba01a3d.pdf

Marchette, D. (2001). Computer Intrusion Detection and Network Monitoring, A Statistical Viewpoint. New York: Springer.

Mirkovic, J., Reiher, P. (2004). A Taxonomy of DDoS Attacks and Defense Mechanisms. ACM Computer Communication Review, 34(2), 39-54. Available at: https://www.princeton.edu/~rblee/ELE572Papers/Fall04Readings/DDoSmirkovic.pdf

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M. (2013). A survey of intrusion detection techniques in Cloud, Journal of Network and Computer Applications, 36(1), 42-57. https://doi.org/10.1016/j.jnca.2012.05.003

Peng, T. (2002). Defending Against Distributed Denial of Service Attack Using Selective Pushback. Proceedings of the Ninth IEEE International Conference on Telecommunications (ICT 2002), Beijing, China.

Porras, P.A., Valdes, A. (1998). Live Traffic Analysis of TCP/IP Gateways. Proceedings of the ISOC Symposium on Network and Distributed System Security (NDSS'98), San Diego. Available at: http://www.csl.sri.com/projects/emerald/live-traffic.html

Provost, F., T. Fawcett. (2001). Robust Classification for Imprecise Environments, Machine Learning, 42(3), 203-231. http://dx.doi.org/10.1023/A:1007601015854

Robertson, S., Siegel, E., Miller, M., Stolfo, S. (2003). Surveillance Detection in High Bandwidth Environments, In Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEX 2003), Washington DC. http://dx.doi.org/10.1109/DISCEX.2003.1194879

Sommestad, T., Hunstad, A. (2013). Intrusion detection and the role of the system administrator. Information Management & Computer Security, 21(1), 30-40. http://dx.doi.org/10.1108/09685221311314400

Staniford, S. Hoagland, J.A., McAlerney, J.M. (2002). Practical Automated Detection of Stealthy Portscans, Journal of Computer Security, 10(1-2), 105-136. Available at: http://hoagland.org/papers/Practical%20automated%20detection%20of%20stealthy%20portscans.pdf

Tavallaee, M., Stakhanova, N., Ghorbani, A. (2010). Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Method. IEEE Transactions on Systems, Man, and Cybernetics: Part C, 40(5), 516-524. http://dx.doi.org/10.1109/TSMCC.2010.2048428

Wang, H., Liu, J. (2008). Combining evidence in multivariate data spaces. Proceedings of the 8th International FLINS Conference on Computational Intelligence in Decision and Control, 11-16, Madrid, Spain.

Wang, W., McClean, S. (2008). Deriving evidence theoretical functions in multivariate data space. IEEE Transactions on Systems, Man, and Cybernetics, 38(2), 455-465. http://dx.doi.org/10.1109/TSMCB.2007.913593

Xu, E., Wermus, M., Bauman, D.B. (2011). Development of an integrated medical supply information system, Enterprise Information Systems, 5(3), 385-399. http://dx.doi.org/10.1080/17517575.2011.566630