Proposal of Model and Message Format for Sharing Information between CSIRTs and IDSs

Fernando A. Pestana Júnior, Zair Abdelouahab, Edson Nascimento

Federal University of Maranhão
CCET/DEEE
Av dos Portugueses, Campus do Bacanga
São Luís – MA, 65080-040, Brazil
tel: +55 98 3217 8832
fax: +55 98 3217 8241
pestanajunior@hotmail.com, {zair,edson}@dee.ufma.br

Abstract. This article proposes a message format for sharing information between Computer Security Incident Response Team (CSIRTs) and Intrusion Detection Systems (IDSs), aiming the achievement of an automatic update of the response actiond data base in IDSs, based on restrictive short-term measures suggested in security alerts issued by CSIRTs. This model is based on Web services and Extensible Markup Language (XML) technologies. It is also presented a data format to these security alerts as an extension of the Common Alerting Protocol.

Keywords. Intrusion Detection, Web services, Extensible Markup Language

1 Introduction

Intrusion Detection Systems (IDSs) are systems composed of software or hardware hich automate the monitoring process of events that occur in a computing system or network, with the aim of identifying signs of security problems [1].

Mechanisms which allow exchanging information between IDSs and Computer Security Incident Response Team (CSIRTs) are necessary. Thus, a better understanding of events surrounding the protected domain, allowing that prevention and response measures are taken.

In this work it is proposed a model for sharing information between CSIRTs and IDSs, aiming to make possible for IDSs to update automatically its response action data base, based on short-term restrictive measures suggested in security alert issued by CSIRTs, using Web services [10] and XML [9] technologies.

The rest of this article is organized this way: Section 2 presents general information about Common Intrusion Detection Framework (CIDF) and Common Alerting Protocol (CAP) which have been used as a basis for this article. In section 3, the architecture model for IDSs, as well as its requirements and functionalities. Section 4 presents a model proposal for sharing information between CSIRTs and IDSs, where a proposal for coding the security alerts in XML is presented. In section 5, the implementation and tests are described; in section 6, the conclusions are presented.

© A. Gelbukh, C. Yáñez Márquez, O. Camacho Nieto (Eds.)
Advances in Artificial Intelligence and Computer Science
Research on Computing Science 14, 2005, pp. 285-296

2 Background

2.1 Common Intrusion Detection Framework (CIDF)

CIDF [4] is a project by Defense Advanced Research Projects Agency (DARPA) and one model that represents didactically the operation of an IDS, showing the flow of information and the basic function of CIDF is shown in Figure 1.

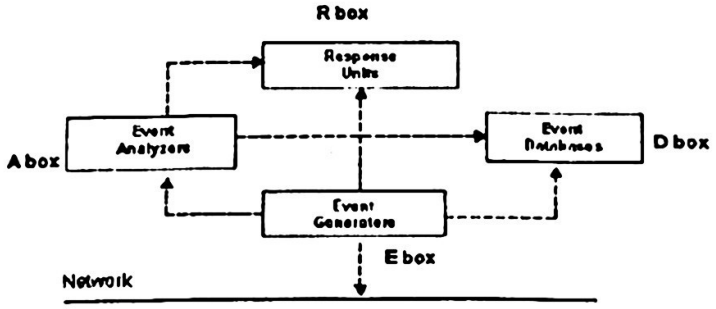

Fig. 1. Common Intrusion Detection Framework with captor of packets in the network

2.2 Common Alerting Protocol (CAP)

CAP [6] is a simple but general format for exchanging all kinds of emergency alerts. These emergency alerts should inform people about imminent danger or threats which can cause some kind of damage. CAP can also be used on all kinds of communication network, including TCP/IP (Transmission Control Protocol/Internet Protocol).

The main function of alert messages in CAP format is to build a unique message which has the capacity of activating all kinds of alerting systems. A secondary function of these messages is to standardize the alerts released by various sources so that can be aggregated and compared assisting the detection of standards.

3 Sharing Information between IDSs and CSIRTS

According to CIDF, the ability of IDSs and their components to share information about security incidents is very important and would allow systems to warn others about possible imminent attacks [4].

However it would not be safe to allow direct access to the components of an IDS through an outer entity and thus the introduction of a information share box in the CIDF architecture would eliminate this direct access and could supply the following functions necessary to secure this type of application: digital encryption and signature. Figure 2 shows the information share box (exchange box) added to the CIDF architecture.

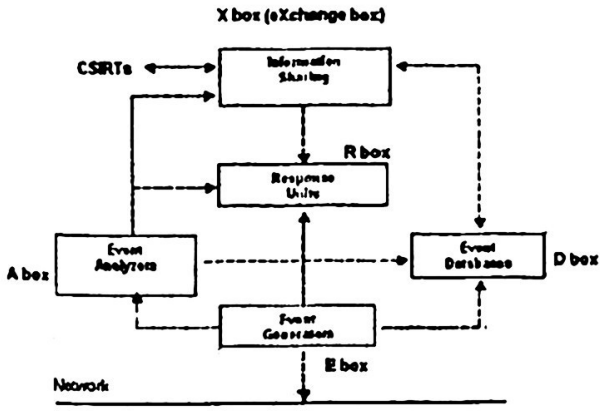

Fig. 2. Information Share Box

The information share box will be accountable for searching and receiving alerts generated by CSIRTs, processing (them) and send its information to the data base components where the response actions data base will be updated.

In case a new threat or vulnerability is detected, the CSIRT release a security alert, using a standardized format. IDSs monitor constantly the CSIRTs and as soon as a new security alert is available, the information share box will recuperate it, update the response actions database and inform the administrator about possible and necessary updates to the existing systems.

3.1 Proposal of Application for the Architecture Presented

In Figure 3. there is a set of domains monitored by different IDSs.

The information share may be added to existing IDSs, in form of a new box. As soon as this box is started the domain is integrated to the group of domains which share security information with a CSIRT.

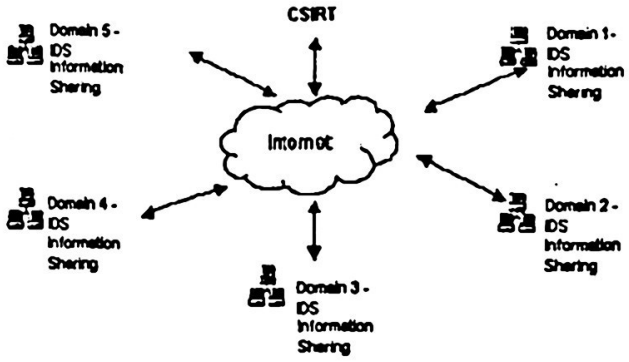

Fig. 3. System architecture

In the example of Figure 3, all domains may send incidents information to CSIRT which will process them searching for attacks patterns or new vulnerabilities. If a threat is detected based on analysis of these informations, the CSIRT will generate a security alert which will be available to all domains.

The information share box will be monitoring the CSIRT constantly searching for security alerts and as soon as it perceives that a new alert is released, this alert will be recuperated and its information can be used to update the IDS response actions data base .

In the next section we will specify a standard format for communication messages between CSIRTs and IDSs.

4 Proposal of a model for information share between CSIRTs and IDSs.

In this section a model for information share between CSIRTs and IDSs is proposed, with the objective of enabling IDSs to automatically update its set of response actions, based on short-term restrictive measures suggested in security alerts released by CSIRTs using Web services and XML technologies.

4.1 Response Actions Data Base (RADB)

The RADB contains information about response actions which are to be taken according to the attack detected. The scheme of RADB is a representation of the standard format, proposed to the alerts released by CSIRTs which will be described in this section.

A response may be defined as a plan of action which the system has when an intrusion is detected [2] and this plan of actions will constitute the RADB. Below the main response techniques are listed [7]: generate reports, alarms, cancel jobs and cancel user's session, investigate suspects, blocking IP addresses, disconnect the host, use a additional intrusion detection tool, disable ports and services affected, investigate the connection, create backup and use temporary file for protection.

4.2 Updating RADB based on Alerts released by CSIRTs

According to Figure 4, a Central Security Agency will be responsible for collecting alerts released by several CSIRTs, which will have information about new vulnerabilities detected and they will be available in some standardized format. This share of information by CSIRT, apart from the current pages in HyperText Markup Language (HTML), it may be done by means of Web services and XML which would be consulted constantly by the Central Security Agency searching for new alerts.

As long as the Central Security Agency receives a new alert released by a CSIRT, it will be processed and its information will be stored in the RADB.

In figure 4, there are some CSIRTs which emit security alerts and could be used as information source to update the RADB.

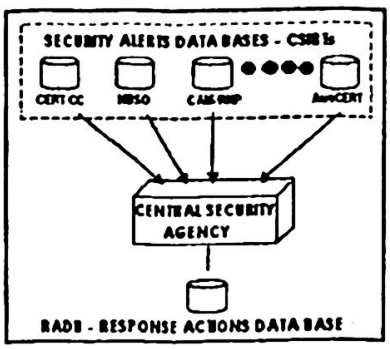

Fig. 4. Model for update automatically the response action data base

With the alerts being published in HTML, the security administrator must read all alerts released and verify which vulnerabilities published are applicable in his environment and take the suggested measures in order to stop an intruder's access to compromised systems, limit the extent of an intrusion, and prevent an intruder from causing further damage [3].

These restrictive short-term measures are exactly the ones which can feed the RADB, keeping the IDS response mechanism updated and able to respond to security incidents which involve a new vulnerability described in the alert released by CSIRT. In order to make it possible is necessary that the attack detection mechanism is also updated. Updating the attack detection mechanism is out of the scope of this work.

In order to accomplish the information share between CSIRTs and IDSs it is necessary to have an agreement between them about the set of elements which they will use and know what these elements mean, and what Web services methods they are going to use, what is the function of these methods and in which order they are called when more than one method is necessary.

Apart from these circumstances it is necessary a standard to the CSIRTs security alerts and this standard must allow automatic processing. With the standardization the IDS will not have to understand and decode several formats. Thus, it is proposed an alert codification format as an extension of CAP.

The structure of this alert message concerning security problems on the Internet based on the structure of the CAP alert message consists of one segment <alert> which can contain one or more segments <restriction> and <info>. The segments <info> will contain one segment <instruction> which in turn may contain one or more segments <applyPatch>, <removePatch>, <blockAccess>, <disableFeature>, <modifyFile> and <PermissionChange>. That structure is shown in figure 5.

The following extensions were inserted in the structure of the CAP alert message:

- The required element "scope" of the CAP message must have its value determined as "Restricted", pointing out that a certain alert is destined to users of specific systems.
- The optional element "restriction" was redesigned as a required segment which will point out which systems an alert refers to. Figure 6 shows that alteration.
- The optional element "instruction" contained in the segment "info" was redesigned as a required segment and will contain suggestions of action to be taken. Figure 7 shows the structure of those elements.

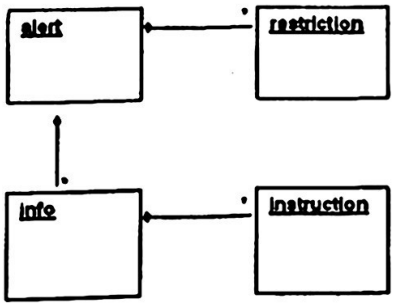

Fig. 5. Structure of the alert message for security problems on the Internet.

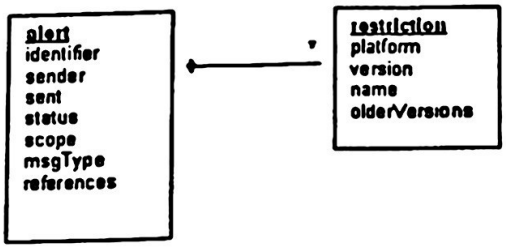

Fig. 6. Structure of segments "alert" and "restriction".

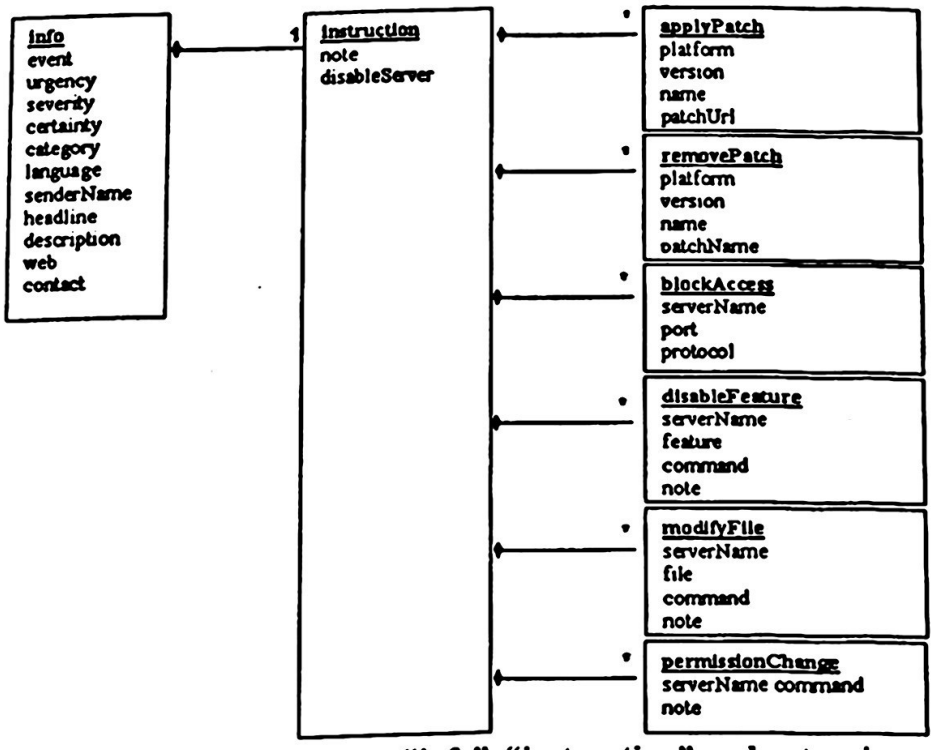

Fig. 7. Structure of elements "info", "instruction" and extensions.

4.3 Data Dictionary

Element Name	Definition and (optionality)	Notes or Value Domain

"restriction" Element and Sub-elements

restriction	The container for all component parts of the restriction sub-element of the alert message. (required)	Multiple occurrences are permitted within a single <alert>. Identifies vulnerable systems.
platform	The code denoting the platform name of the vulnerable system. (optional)	Code Values: "Sparc" – Sun Sparc Platform. "x86" – Intel Platform. "Windows" – Microsoft Windows. "Solaris" – Sun Solaris. "Linux" – Linux Platform.
version	System version identification. (optional)	A number or string uniquely identifying the system version, assigned by the vendor. No spaces or restricted characters (< and &).
name	The identifier of the system name. (required)	A string uniquely identifying the system name, assigned by the vendor. No spaces or restricted characters (< and &).
olderVersions	The code denoting if an alert affects older versions of a specific system. (optional)	Code Values: "Y" - An alert affects all older versions of a specific system. "N" - An alert does not affect all older versions of a specific system.

"instruction" Element and Sub-elements

instruction	The container for all component parts of the instruction sub-element of the info sub-element of the alert message. (required)	Single occurrence permitted within a single <info> block. In addition to the specified sub-elements, may contain one or more <applyPatch>, <removePatch>, <blockAccess>, <disableFeature>, <modifyFile>, <permissionChange> blocks.
note	The text describing the recommended action to be taken by recipients of the alert message.	

	(required)	
disableServer	The group listing of server names to be disabled.	Multiple space-delimited server names may be included. Server names including white space must be enclosed in double-quotes. A string uniquely identifying the server name, assigned by the vendor.

"applyPatch" Element and Sub-elements

applyPatch	The container for all component parts of the applyPatch sub-element of the instruction sub-element of the info sub-element of the alert message. (optional)	Refers to a patch file. Multiple occurrences are permitted within a single <instruction> block.
platform	The code denoting the platform name of the vulnerable system. (optional)	Code Values: "Sparc" – Sun Sparc Platform. "x86" – Intel Platform. "Windows" – Microsoft Windows. "Solaris" – Sun Solaris. "Linux" – Linux Platform.
version	System version identification. (optional)	A number or string uniquely identifying the system version, assigned by the vendor. No spaces or restricted characters (< and &).
name	The identifier of the system name. (required)	A string uniquely identifying the system name, assigned by the vendor. No spaces or restricted characters (< and &).
patchUri	The identifier of the hyperlink for the patch file. (required)	A full absolute URI, typically a Uniform Resource Locator that can be used to retrieve the patch file over the Internet.

"removePatch" Element and Sub-elements

removePatch	The container for all component parts of the removePatch sub-element of the instruction sub-element of the info sub-element of the alert message.	Refers to an older patch that must be removed before install a new patch. Multiple occurrences are permitted within a single <instruction> block.

	(optional)	
platform	The code denoting the platform name of the vulnerable system. (optional)	Code Values: "Sparc" – Sun Sparc Platform. "x86" – Intel Platform. "Windows" – Microsoft Windows. "Solaris" – Sun Solaris. "Linux" – Linux Platform.
version	System version identification. (optional)	A number or string uniquely identifying the system version, assigned by the vendor. No spaces or restricted characters (< and &).
name	The identifier of the system name. (required)	A string uniquely identifying the system name, assigned by the vendor. No spaces or restricted characters (< and &).
patchName	The identifier of the patch name. (required)	A string uniquely identifying the patch name, assigned by the vendor. No spaces or restricted characters (< and &).

"blockAccess" Element and Sub-elements

blockAccess	The container for all component parts of the blockAccess sub-element of the instruction sub-element of the info sub-element of the alert message. (optional)	Refers to ports and protocols that must be blocked. Multiple occurrences are permitted within a single <instruction> block.
serverName	The identifier of the server where ports or protocols will be blocked. (required)	A string uniquely identifying the server name, assigned by the vendor. No spaces or restricted characters (< and &).
port	The integer from 0 through 65535 indicating the port number to be blocked. (required)	Port numbers assigned by IANA (Internet Assigned Numbers Authority).
protocol	The code denoting the protocol name. (required)	Code Values: "Tcp" – Transmission Control Protocol. "Udp" – User Datagram Protocol.

"disableService" Element and Sub-elements

disableService	The container for all component parts of the disableService sub-element of the instruction sub-	Identifies a service or feature of this service that must be disabled.

	element of the info sub-element of the alert message. (optional)	Multiple occurrences are permitted within a single <instruction> block.
serverName	The identifier of the server where a service will be disabled. (required)	A string uniquely identifying the server name, assigned by the vendor. No spaces or restricted characters (< and &).
service	The identifier of the service that will be disabled. (required)	A string uniquely identifying the service or a feature of this service, assigned by the vendor. No spaces or restricted characters (< and &).
command	The text describing a command and its options to disable a service or a feature of this service.	
note	The text describing the service or a feature of this service to be disabled and probably impacts of these actions.	

"modifyFile" Element and Sub-elements

modifyFile	The container for all component parts of the modifyFile sub-element of the instruction sub-element of the info sub-element of the alert message. (optional)	Refers to a file name with its full path. Typically a configuration file. Multiple occurrences are permitted within a single <instruction> block.
serverName	The identifier of the server where a file or files will be modified. (required)	A string uniquely identifying the server name, assigned by the vendor. No spaces or restricted characters (< and &).
file	File name with its full path. (required)	
command	Text describing the necessary command to modify a file. (optional)	Multiple occurrences are permitted within a single <modifyFile> block.
note	Text describing the command and probably impacts of its actions. (optional)	

"permissionChange" Element and Sub-elements

permissionChange	The container for all component parts of the permissionChange sub-element of the instruction sub-element of the info sub-element of the alert message. (optional)	Used to modify directory or file access permission. Multiple occurrences are permitted within a single <instruction> block.
serverName	The identifier of the server where directory or file access permission will be modified. (required)	A string uniquely identifying the server name, assigned by the vendor. No spaces or restricted characters (< and &).
command	Text describing the necessary command to modify a user permission to access directories or files. (required)	Multiple occurrences are permitted within a single <permissionChange> block.
note	Text describing the command and probably impacts of its actions. (optional)	

5 NIDIA System

NIDIA is a proposal of a multiagent [8] IDS, able to analyze data from hosts logs and network traffic packets to generate an attack suspicion level to the protected network.

The NIDIA architecture is inspired in the CIDF logical model, and has agents acting as event generators (sensor agents), data analysis mechanisms (system monitoring and security evaluation agents) and response module (system controller agent). Apart from that, there are agents accountable for system integrity (system integrity agent) and system update (system update agent).

There are also databases to store its security policy strategy (STDB), the response actions to be taken in case of suspicious activity (RADB), intrusion and intruder patterns (IIDB) and a database to store intrusion data (DFDB).

Aiming to keep the response actions database updated, the System Updating Agent was implemented in accordance with the multiagent architecture proposed in [5] for NIDIA System. This agent will work as Central Security Agency, as described in section 4.2.

In order to test, a prototype representing a CSIRT was implemented, which makes available through web service several security alerts. A database was developed for this prototype and it was constantly updated with CERT®/CC vulnerability notes released in 2004.

Through these implementations was possible to maintain updated the NIDIA response actions data base. The System Updating Agent constantly sent requests to the

prototype of CSRIT which processes them and in case there was a new alert, it was sent to NIDIA in XML format.

After that, intrusion tests were made on the network protected by NIDIA. As soon as the invasions were detected, the system controller agent responded according to the data updated and stored in RADB.

6 Conclusion

With daily discovery of new threats and vulnerabilities in computing systems, updating IDS becomes a constant worry.

This work proposes to inform how to keep the Response Actions Data Base (RADB) updated through information share between IDSs and CSIRTs.

However, in order to occur this information share it is necessary a standard to format the alert data , which was proposed as an extension of Common Alerting Protocol (CAP).

Using prototypes, based on society of intelligent agents, Web services and Extensible Markup Language – XML technologies, it was possible establish information share and keep the RADB update, which proves viability of the model herein presented.

References

1. Bace, Rebecca; Mell, Peter. Intrusion Detection Systems. NIST Special Publication, SP800-31, (1999).
2. Bace, Rebecca Gurley. Intrusion Detection. Macmillan Technical Publishing, (1999).
3. Carnegie Mellon University / Software Engineering Institute. Responding to Intrusions. CMU/SEI-SIM-006, February (1999).
4. CIDF Working Group. The Common Intrusion Detection Framework Architecture. Draft CIDF (1998).
5. Lima, C. F. L. et al. The NIDIA Project Network Intrusion Detection System based on Intelligent Agents. Proceedings of Tenth Latin-Ibero-American Congress on Operations Research and Systems. Mexico City, Mexico (2000), pp. 212-217.
6. Organization for the Advancement of Structured Information Standards. Common Alerting Protocol, v. 1.0. OASIS Standard 200402 (2004).
7. Santos, G. de L. F.; Nascimento, E.. An Automated Response Approach for Intrusion Detection Security Enhancement. Proceedings of VII International Conference on Software Engineering and Applications. Marina Del Rey, California, USA (2003).
8. Weiss, Gerhard; Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence, The MIT Press - Cambridge, Massachusetts, London, England (1999).
9. World Wide Web Consortium. Extensible Markup Language (XML) 1.1. April (2004).
10. World Wide Web Consortium. Web Services Architecture. February (2004).