AN ATTACK SIGNATURE MODEL TO COMPUTER SECURITY INTRUSION DETECTION

Adriano M. Cansian
Artur R. A. da Silva
Marcelo de Souza

ACME! Computer Security Research Lab.
UNESP - São Paulo State University
São José do Rio Preto, SP – Brazil
http://www.acme-ids.org

ABSTRACT

Internal and external computer network attacks or security threats occur according to standards and follow a set of subsequent steps, allowing to establish profiles or patterns. This well-known behavior is the basis of signature analysis intrusion detection systems. This work presents a new attack signature model to be applied on network-based intrusion detection systems engines. The AISF (ACME! Intrusion Signature Format) model is built upon XML technology and works on intrusion signatures handling and analysis, from storage to manipulation. Using this new model, the process of storing and analyzing information about intrusion signatures for further use by an IDS become a less difficult and standardized process.

INTRODUCTION

A great number of factors contribute to grant computer system access to aggressors, eventually resulting on further private information stealing or other risks. Some type of security disruption has already occurred in the majority of computers, causing external aggressors (or still legitimate users) to obtain some sort of unauthorized access. Even a supposed safety system may be vulnerable to legitimate users abusing their privileges, or being compromised by improper practices. Inside this scenario, once the attack is considered inevitable, must be used some mechanisms to detect and protect the system against aggressors trying to penetrate or disrupt it, or even legitimate users making bad use of their privileges.

Nowadays, one of the most used security mechanisms is the firewall system, responsible for protecting threats inside from outside network. But a firewall is not a complete tool, due to it is not able to work on the internal side of the network. Neither it is able to stop an intrusion attempt or privileged misuse, for which it has not been previously programmed.

So, a suitable approach is to use a system able to protect the internal network with an adaptive behavior, which can learn new patterns of actions to identify an attacker. It is also desirable that this tool, once with the capacity to detect malicious actions, takes some prevention or counter-measures decisions about the situation, based on the knowledge given to it.

A good tool could be a network-based intrusion detection system (IDS). In a nutshell, intrusion detection systems do exactly as the name suggests: they detect possible intrusions. More specifically, IDS tools aim to detect computer attacks and/or computer misuse, and to alert the proper humans upon detection. It is desired that an IDS makes use of some computing intelligence algorithm, able to provide power to learn when an intrusion attempt is detected, or when an intrusion is already in progress, to react against it.

The classical intrusion detection approach is based on tracking or observation of profiles and patterns, whose are present in any intrusive behavior. Such patterns consist an "intrusion signature", the IDS main information resource. Under this situation, it is possible to affirm that an IDS functionality can be enhanced when the intrusion signatures handling mechanism is improved.

Upon the before-mentioned discussion, this paper presents a new intrusion signature representation model, named AISF (ACME! Intrusion Signature Format). The main goal of AISF design is to achieve a clear definition of intrusion signatures coding, storage and processing. Thus, any computer security analyst or team, or an intrusion detection related entity or person,

0-7803-7625-0/02/$17.00 ©2002 IEEE.

can commit to an attack signatures reporting and analysis standard.

NETWORK-BASED INTRUSION DETECTION AND INTRUSION SIGNATURES

Intrusion detection is defined in [1] as "the problem of identifying people who are using a computational system without authorization (crackers for example) and those who have legitimate access to the system but are abusing their privileges", while intrusion is "any group of actions that try to compromise the integrity, privacy or availability of a resource". In this context, IDS can be defined as software or hardware mechanisms in which the process of monitoring events happening in a computer or network is automatic. This process is made by analyzing events in search for signs that represent possible security treats (more specifically, intrusion signs). The functionalities of a detection system become of vital importance as they supply means to infer over the content of allowed connections, and to detect the ones which present a suspicious behavior or are in disagreement with the local security policies.

Current approaches to detecting intrusions can be broadly classified into two categories: **anomaly detection** and **misuse detection** [2]. **Anomaly detection** is based on the premise that, often, intrusive activity manifests itself as an abnormality. The usual approach here is to devise metrics indicative of intrusive activity, and detect statistically large variances on these metrics. Examples might be an unusually high number of network connections within an interval of time, unusually high CPU activity, or use of peripheral devices not normally used. This approach has been studied extensively and implemented in a large number of systems. It attempts to quantify the acceptable behavior and thus identify abnormal behavior as intrusive. **Misuse detection** attempts to encode knowledge about attacks as well defined patterns and monitors for the occurrence of these patterns. For instance, exploitation of the *fingerd* and *sendmail* bugs, used in the Internet Worm attack [3], would be in this category. This technique specifically represents knowledge about unacceptable behavior and attempts to detect its occurrence, once that this behavior can be described by a specific pattern or sequence of events and its data (the so-called intrusion signatures).

Another determined classification consists of separating IDSs according to treatment performed on data. These IDSs are classified on **host-based** and **network-based** systems. The first one analyzes audit data usually picked up by the operating systems, looking for the attack patterns in several ways, like file integrity checking. **Network-based** systems examine the network flow being monitored online ("sniffing" it from the network). The present work will concentrate in this last IDSs flavor. A lot of helpful information source on existing or under development IDSs is available [4]. One of the IDSs that should be granted special attention is ACME-IDS, designed with a special model which uses neural network mechanisms for intrusion detection [5] [6]. An intrusion signature is a specification of aspects, conditions, arrangements and inter-relationships between events depicting an attack, a breaking of access, another type of abuse or any attempt of these. An intrusion pattern has the same meaning as an intrusion signature.

AISF DESIGN

AISF can be shortly defined as a proposal for standardizing the coding, storage, processing, sharing, exchanging and reporting intrusion patterns, characteristics of misuse behavior against computer network environments, in such a way that they can be used freely among the systems and entities involved in detection of attacks. Technically speaking, AISF (ACME! Intrusion Signature Format) is a data structure comprised of an independent set of modules, which contains intrinsically related information that allows the accurate and concise reporting of attack signatures, describing from descriptive characteristics to implicit details of network protocols. Its organization is based on XML specification [7], which supplies after all, simplicity, adaptability, high portability, flexible use and maintenance. XML allows a specific markup to be created, defining ideas and sharing them, which characterizes exactly the feature needed by AISF to unify and disclose the signature patterns. An AISF instance will be referred to from now on as "**AIS**", or AISF object, which will consist of a properly formatted XML document containing important data of an intrusive event. For each event there will be only one AISF object, which, however, can be revised and result in new versions. AISF is initially used to facilitate the way attack signatures are stored and processed by an IDS. Several AISF objects are deposited in a global database, maintained by a reliable entity that wants to control and use these instances, allowing the addition of new AISs about

new attacks. Using AISF, an IDS will have a standard signatures management system, capable of processing the model starting with a parse of XML documents in the database.

Another employment for AISF is the possibility of being used inside less complex systems, taking into account signature analysis, like a simple scan detector, or a network dump analyzer, generated by a tcpdump-like "sniffer" [8], comparing these dumping with the AISF objects. Alert reporting can be highly facilitated, once AISs are generated with descriptive texts, besides the own useful contents of signatures.

Extensive work has been done in the subject of signature handling, but without much success, since the existing systems do not focus on the standardization as a substantial objective. Examples of this situation are the arachNIDS [9] and Snort Signatures Database [10]. Besides them, it is worth mentioning the efforts of IDWG-IETF [11] in standardizing the exchange of information among IDS elements, but no effort concerning the signatures as the real matter. Following, is presented detailed AISF data structure, through the description of existing modules and its components. It is important to notice that modules described here serve for visualization purposes only, since the specification itself follows XML patterns, as explained below.

The first and most important AISF module consists of the **Signature Identification Module** (figure 1). It is responsible for the identification of an event, as well as some details about the model itself.

Version
ID
Name
Serial Number
Credits
Next Module

Figure 1 – *Signature Identification Module*

This initial module is the only one that should obligatorily contain all of its fields properly filled. Below there is a list of the fields and its functions:

Version: field devised for the identification of the AISF model. At the time of writing, an initial version of 0.0.7 will be admitted;

ID: identifies the attack described by the AIS. It is a unique number given by the model administrative entity;
Name: common name of the intrusion event;
Serial Number: number that describes the AIS version, based on the generation date (yyyymmdd-xx);
Credits: AIS authors (entity or person);
Next Module: identifies the next module of the AIS;

Note that the **Next Module** fields must contain the name of the subsequent data module. If this one does not exist, the field should be left blank.

The second module (figure 2), **Signature Information Module,** is responsible for more descriptive information about the attack. This module, as well as the following one, is optional.

Module Length: describes how many fields (including the **Next Module** field) are to come with this module. With this, a simple parsing system can ignore a module if it is not understandable or needed;
Security Level: number (from 0 to 100) describing how dangerous this event can be;
Category: intrusion event category, like scan, DoS, or interactive attack;
Description: AIS event description;
Other IDs: reference IDs for this attack, like CVE [12] or BugTraq ID [13];
Impact: what is the consequence of this event;
Attack Scenario: what is needed for this attack to happen;
Target System: systems that are more commonly affected by this intrusion;
Next Module: the same as described above;

Module Length
Security Level
Category
Description
Other IDs
Impact
Attack Scenario
Target System
Next Module

Figure 2 – *Signature Information Module*

The third module (figure 3) pertaining to the informative modules class is the **Signature Characteristics Module**, which can report some attack inherent information.

Module Length
Ease of Attack
False Positive Level
False Negative Level
Recommended Actions
Next Module

Figure 3 – *Signature Characteristics Module*

Module Length: the same as above;
Ease of Attack: number (from 0 to 100) describing how easily this attack can be realized;
False Positive Level: approximation percentage of false positives this event can trigger;
False Negative Level: approximation percentage of false negatives this event can trigger;
Recommended Actions: recommended preventive and/or corrective actions to take;
Next module: as described above;

The following modules are easily understandable, representing the more technical side of an attack signature. They represent required information for intrusion detection, like data link, network and transport layers data, besides the payload of the session. In the present version of AISF, there are just the more acquainted protocols of these layers, like Ethernet, IP, ICMP, TCP and UDP. Nothing interferes in the addition of other protocols, like PPP, IPv6, application protocols (RPC, HTTP) [14], among others.

The next module, **Data Link Protocols Module** (figure 4), refers to Ethernet data link protocol data.

Name	Description
Module Length	Number of module fields
Source Address	Source MAC address
Destination Address	Destination MAC address
Next Module	Identifies the next AISF module

Figure 4 – *Data Link Protocols Module for Ethernet*

Next, there is the Network Protocols Module (figure 5), referring to inherent network layer data, and in the case of the TCP/IP suite, the IPv4 protocol.

Name	Description
Module Length	Number of module fields
Type of Service	IP packet type of service
Fragment ID	Fragment identification number
Flags	IP protocol flags
Fragment Offset	Packet fragment offset
TTL	IP packet time-to-live
Source Address	IP source address
Destination Address	IP destination address
Options	Packet options
Next Module	Identifies the next AISF module

Figure 5 – *Network Protocols Module for IPv4*

The following three tables (figures 6 to 8) demonstrate how AISF can be used to keep data regarding transport and control layers, trough the **Transport and Control Protocols Module**. TCP, UDP and ICMP protocols header information used in the attack can be described by these modules.

Name	Description
Module Length	Number of module fields
Source Port	TCP source port
Destination Port	TCP destination port
Sequence Number	Packet sequence number
Acknowledge Number	Packet acknowledge number
Data Offset	Packet data offset
Flags	TCP flags
Window	Window size
Urgent Pointer	Urgent pointer
Options	TCP options
Next Module	Identifies the next AISF module

Figure 6 – *Transport and Control Protocols Module for TCP*

Name	Description
Module Length	Number of module fields
Source Port	UDP source port
Destination Port	UDP destination port
Next Module	Identifies the next AISF module

Figure 7 – *Transport and Control Protocols Module for UDP*

Name	Description
Module Length	Number of module fields
Type	ICMP Type
Code	ICMP Code
ID	ICMP ID
Sequence	ICMP Sequence number
Next Module	Identifies the next AISF module

Figure 8 – *Transport and Control Protocols Module for ICMP*

AISF **Payload Information Module** (figure 9) is one of the most important. There can be stored information regarding the attack session packets payload. It is considerably useful to describe the interactive attacks, like the ones that explore "buffer overflows", where a great number of "NO-OP" instructions can be found [15]. Thus, with this module, one can describe the payload size, where exactly the important data is found in this payload and other details that aid in the search for intrusion tokens. Still in this module, on the **Contents** field, markup tags are defined to facilitate string matching, like case sensitiveness and data order.

Name	Description
Header Length	Number of header fields
Size	Payload size
Offset	Payload offset to start pattern matching
Depth	How far to search into the packet
Contents	Payload contents
Next Module	Identifies the next AISF module

Figure 9 – *Payload Information Module*

Notice that although currently there are no other subsequent modules, the **Next Module** field has been reserved in the **Payload Information Module**, to aim further specific modules.

AISF AND ITS XML DEFINITION

The XML (Extensible Markup Language) specification allows AISF to be written with a simple but intelligent character. In this way, it becomes possible to modulate the outline, assuring that any system can interpret it. With the use of XML tags, it is feasible to easily define the previously explained data structure. Another positive aspect of using XML is the occurrence of pre-validation when new models are defined. With the built-in DTD (Data Type Definition) feature, the initial model could be specified, allowing further extensions or improvements according to future needs. It is important to point out that due to XML inherent portability, the function libraries and pre-existing APIs can be used to accomplish posterior document parsing, being easily embedded into IDSs, for example. API standards like DOM (Document Object Model) and SAX (Simple API for XML) are globally accepted, besides the libXML library under GNU/Linux environment. AISF systems can use **XSL** (Extensible Style Language) and **CSS** (Cascade Style Sheets) for enhanced stylized visualizations, without requirements of complex applications or applets to make format conversions.

FINAL CONSIDERATIONS

This work showed a standard intrusion signature representation model using the XML specification. Throughout the article, the importance of intrusion signatures and its use by intrusion detection systems, like ACME!-IDS, was presented. Special attention has been devised to the need for a unified way of storing, processing, analyzing and reporting intrusion patterns.

Besides that, has been also showed the modularization of the model as a very important feature. It provides possibility for different systems to share information related to intrusive events. This modularization also makes easy the parsing of AISs, diminishing the process overhead, which is a good desirable feature of IDSs. The use of XML comes with a new standardization tendency regarding information sharing. It supplies pre-processing of data, organization uniformity and easiness of graphical signature representation.

Proposals for future works include the specification of modules responsible for those attacks solely based on multi-packet techniques, like "flooding attacks" and DDoS. It is important to mention that this feature was not focused on as the main purpose in this paper, but a misuse-based model. The development of interfaces for XML data input, along with high performance database retrievers and parsers, can also be proposed.

ACKNOWLEDGMENTS

This work is partially supported by FAPESP – Fundação de Amparo à Pesquisa do Estado de S.Paulo, Brazil. The authors would like to thank to Sergio Leugi Filho and Jarbas de Freitas Peixoto to important contributions about this research.

REFERENCES

[1] Spafford, E.; Balasubramaniyan, J.S.; Fernandez, J.O.G.; Isacoff, D. and Zamboni, D. *An architecture for intrusion detection using autonomous agents.* COAST technical Report 98/05, 1998.

[2] Stephen E. S. *Tools For Misuse Detection.* In Proceedings of ISSA, 1993.

[3] Spafford E. *The Internet Worm Report.* Technical Report 823 - Purdue University, 1990.

[4] Sobirey, M. *Michael Sobirey's Intrusion Detection Systems page,* 2000. http://www.nks.informatik.tucottbus.de/%7esobirey/ids.html (last seen January 27, 2002)

[5] Cansian, A.M.; Moreira, E.M.; Carvalho, A.C.P.L. and Bonifácio Jr., J.M. *Network Intrusion detection using neural networks.* In: Proceedings of International Conference on Computational Intelligence and Multimedia Applications, ICCIMA'97. Gold Coast, Australia: 1997. pages 276-280.

[6] Bonifácio Jr. J.M.; Cansian, A.M.; Carvalho, A.C.P.L. and Moreira, E.M. *Neural networks applied in intrusion detection systems.* In: Proceedings of IEEE International Joint Conference on Neural Networks – IJCNN'98. Anchorage, Alaska: IEEE, 1998.

[7] World Wide Web Consortium *XML (Extensible Markup Language) 1.0.* October 6, 2000. http://www.w3.org/TR/2000/REC-xml-20001006 - (Last seen Jan 28, 2002)

[8] Tcpdump Project . http://www.tcpdump.org/

[9] Whitehats Inc. *arachNIDS – The Intrusion Event Database.* http://www.whitehats.com (last seen December 10, 2001).

[10] Roesch, M. *Snort Signatures Database.* http://www.snort.org/snort-db/ (last seen January 28, 2002).

[11] Internet Engineering Task Force – Intrusion Detection Work Group, *Intrusion Detection Exchange Format. http://www.ietf.org/html.charters/idwg-charter.html (last seen January 28, 2002).*

[12] MITRE Corporation. *Common Vulnerabilities and Exposures.* 2001. http://cve.mitre.org/ (last seen January 28, 2002)

[13] BugtraqID http://www.securityfocus.com (last seen January 28, 2002)

[14] Stevens, W. R. *TCP/IP Illustrated – Volume 1 – The Protocols.* Addison-Wesley Professional Computing, 1994

[15] Cooper, M., Fearnow, M., Frederick, K. and Northcutt, S. *Intrusion Signatures and Analysis.* 1st Edition - New Riders Publishing, January 2001, 408 pages.