Testing the efficacy of an intrusion signature representation model

EURÍPEDES L. L. JÚNIOR, LUCIANO B. DE PAULA, ADRIANO M. CANSIAN
UNESP – Universidade Estadual Paulista
ACME! Computer Security Research Lab.
15055-000, São José do Rio Preto, SP
BRAZIL
{euripedes, luciano, adriano}@acme-ids.org – http://www.acme-ids.org

Abstract: - Nowadays, the computer network security is very important and became a crucial point in every computational systems. For this reason, a robust and efficient security system is necessary. The intrusion detection systems (IDS) largely used today are based on rules. These rules are written after the attack is known, which implies that the more attacks are known, more rules can be built.With more rules, IDS become more efficient. The goal of this paper is to test the efficacy of a new signature representation model – AISF. This model has as feature the ease of information exchange between several IDS, using the XML technology. In this paper will be shown how some signatures were modeled using AISF specification and how it is possible to use it to feed an IDS.

Key–Words: security, intrusion, signature, model, IDS, XML.

1 Introduction

Nowadays, the security of computer networks is more important than it was in the past, and it is less important than it will be in the future. This is due to the growing number of crucial activities that are being developed under computational environments, like e-commerce.

As the importance increases, so does the need for integrity and privacy of those computational system grows. The major priority of the security analysts is how to keep the system free from "holes" that can spoil the security and harm the system.

A tool largely used to protect computational systems is called IDS (intrusion detection system). An IDS is a program that works analyzing the data flow in a network, and tries to identify activities that can be classified as suspicious or offensive. The major part of IDS is based on rules. It means that, if an attack has a signature and this signature is known, a rule to block this kind of event can be built.

If the number of known signatures grows, so does the chance of protecting the victim against this kind of attack. On the other hand, the number of new attacks that is reported every day is very impressive.

The AISF [1] model proposes a standard form to codify those attack signatures, looking for the ease of exporting and importing these data between different IDS.

AISF consists of several modules, using the XML technology, a feature that allows a great portability and flexibility regarding the codified data.

The goal of this paper is to prove the efficacy of this model.

2 The AISF Model

From a technical point of view, AISF is a data structure based on an independent set of modules containing information that reports from informative data of the event to implicit details of network protocols.

AISF organization is based on the XML specification[2], which supplies after all, simplicity, adaptability, high portability, flexible use and maintenance. XML allows a markup to be created, defining the information and sharing them, which is exactly the goal of AISF. The description of the modules can be seen as follows:

1 – Signature Identification Module: this module has the information about the AISF version for this attack codified, as well as its popular name and who codified it;

2 – Signature Information Module: here can be inserted the information about the attack's category (scan, dos, overflow, etc), the conditions that the attack may happen, target systems, a security level (a number from 0 to 100) that describes the danger of the attack and other references for it, like [3][4].

3 – Signature Characteristics Module: holds the information about the false positives and false negatives rate, a number (0 to 100) that indicates the ease to perform the attack, and a recommended action to be done.
4 – Data Link Protocols Module: this module holds the data about the data link protocols, as Ethernet, like for example, the MAC adress.
5 – Network Protocols Module: in this module, the features of the network protocols are codified. The most common is the IP protocol;
6 – Transport and Control Protocol Module: here we can find information about transport and control protocol, like TCP, UDP and ICMP;
7 – Payload Information Module: the comments about the data carried by the packet are written here. This module is very important when the attack is determinated by the string that was present in the packet sent to the victim.

Every module has a field called *Module Length* which contains the number of fields that are present in that module.

3 Testing the AISF model

The model, in order to be efficient, has to allow the information about different types of signatures to be modeled into it. For this reason, a great number of signatures (around 250) were codified into AISF, trying to test its flexibility when holding different kinds of information from them. Were used the data base from AracNIDS[5] and Snort[6], as well as some logs from simulated attacks.

Several types of signatures were codified, ranging from information gathering attempt to real intrusion activities. Several types of services have been reported as well as several types of attacks and platatorms, including: X11, DNS, SSH, IMAP, Netscape Client, Java scripts, LPR, Web – Frontpage, Web IIS, SMTP, SMNP, TFTP, RPC, Netbios, signature involving shellcode, DdoS, DoS, Finger, FTP, Rservice, scans, Telnet, Trojan.

The protocols TCP, UDP and ICMP were involved in this signatures, showing that the AISF model supports any of them.

After all signatures were codified, every relevant information about the attacks could be stored in an AISF format, proving its efficiency when holding this kind of data.

The next section shows examples of attacks codified in AISF format.

4 Codifying a signature using AISF

The example is a log that contains a packet that is tipical of buffer overflow attacks. For example, the wu-ftpd 2.6.0 is a common FTP server that is vulnerable to a very serious remote attack in the SITE EXEC implementation. This attack can be identified by the long string of x86 NOPs (no operation command in Assembly language, code 0x90) instructions [7]. This signature is critical to the remote exploit of the ftpd service (port 21).

See the figure below:

```
----------------------------------
04/28/02-17:13:15.131234 source:1658 -
> destiny:21
TCP TTL:64 TOS:0x0 ID:38851 IpLen:20
DgmLen:457 DF
***AP*** Seq: 0x9164136A  Ack:
0x31DF9933  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 2605235
17971831
0x0000: 50 41 53 53 20 90 90 90 90 90
90 90 90 90 90  PASS ..........
0x0010: 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90  ..............
0x0020: 90 90 90 90 90 90 90 90 90 90
90 90 90 90 90  ..............
```

Fig. 1 – A typical buffer overflow log

Due to space limitations, the figure 1 does not have the full packet.

See this attack codified in AISF:

Signature Identification Module
Version: 0.0.7
ID: 2002-34
Name: FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow
Serial Number: 20020526 - 02
Credits: Eurípedes Laurindo Lopes Júnior and Luciano Bernardes de Paula.
Next Module: Signature Information Module

Signature Information Module,
Module Length: 7
Security Level: 90
Category: Buffer overflow
Description: Wuftpd buffer overflow vulnerability.
Other IDs: CVE CAN200-0574, Bugtraq 1387, AracNIDS 287
Impact: Attempted to gain admnistrator privileges.
Attack Scenario: Red Hat 6.2 system running a wuftpd server 2.6.0(1).
Target System: Linux
Next Module: Signature Characteristics Module

Signature Characteristics Module
Module Length: 6
Ease of Attack: 50
False Positive Level: 20
False Negative Level: 1
Recommended Actions: Upgrade wuftpd version.
Next Module: Network Protocols Module

Network Protocols Module,
Module Length: 10
Type of Service:
Fragment ID:
Flags:
Fragment Offset:
TTL:
Source Address: External
Destination Address: Internal
Options:
Next Module: Transport and Control Protocols Module

Transport and Control Protocols Module
Module Length: 11
Source Port: any
Destination Port: 21
Sequence Number:
Acknowledge Number:
Data Offset:
Flags: ACK
Window:
Urgent Pointer:
Options:
Next Module: Contents

Contents (Conteúdo).
Header Length: 6
Size:
Offset: 0 byte
Depth:
Contents: "|90 90 90 90 90 90 90 90|"
Next Module: NULL

In an attempt to simplify the example, the signature was not codified in XML. These information, in practice, will be coded under XML specifications.

An idea that could be implemented is the creation of an attack signature data base, taking advantage of XML features. As the XML has a great portability, automatic mechanisms could be created to build rules for a specific IDS. These mechanisms, having access to the AISF data base, could extract the information needed by this specific IDS and generate rules to insert into it. The next subsections, shows two IDS as examples, Snort and ACME!-IDS [8] and how an AISF data base could be used with them.

4.1 Using Snort with AISF

Snort is a lightweight network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP networks. Snort is widely used around the world. It uses a flexible rules language to describe traffic that it should collect or pass.

The Snort IDS can have a simple rule to detect this attack. The rule would be:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow Linux"; flags:A+; flow:to_server; content: "|90 90 90 90 90 90 90|");
```

Fig. 2 – A Snort rule

Notice that all information needed to build this rule can be found in the AISF version of the attack. The *External* and the *Internal* adresses can be obtained from the *Network Protocols Module*, the destination port (21) and the protocol flag can be read from the fields *Destination Port* and *Flags,* respectively, from *Transport and Protocols Module.* The message can be taken from the field *Name* of the *Signature Identification Module*. The important string that characterizes the attack ("|90 90 90 90 90 90 90 90 90 90|"), can be extracted from the field *Contents* of the *Contents Module*.

4.2 Using ACME!-IDS with AISF

ACME!-IDS is an intrusion detection system based on neural network developed by the ACME! Computer Security laboratory. See a very brief description of ACME!'s modules:

- *Capture Module*: a module that works like a sniffer, capturing and identifying the network traffic.

- *Pre-selection and Inference Module*: this is the module that judges if a connection is suspicious or not;

- *Connection Module*: this module receives all packets from the suspicious connection and passes its contents to the *Semantic Analyser*;

- *Semantic Analyser*: look for well known strings into the contents of the packets from the suspicious connection. Each well known string is related with a binary number. The set of these binary numbers goes through the neural network, that analyses it, and returns an answer, that indicates if it is an attack or a normal connection.

An AISF data base can be used by ACME!-IDS in two different processes: generation of two pre-filtering rules to be inserted in the *Capture Module* (ACME!-IDS rules, similar to a Snort rule), and the creation of binary data (ACME!-IDS Binary Intrusion String - ABIS) for the *Semantic Analyser*.

The figure 3 contains the structure of how an AISF data base can improve the ACME!-IDS efficiency.

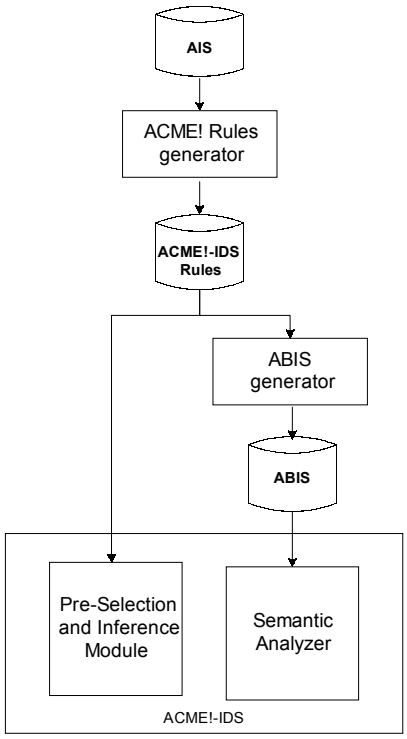

Fig. 3 - ACME!-IDS architecture with AISF

5 Points to be discussed

One of the AISF features is its openess to improvement. If a new attack appears, new fields or modules can be created to support it, if necessary. Some point to be discussed about the current version:

- There are some fields, like *Security level* (from *Signature Information Module*), *Ease of attack, False positive level, False negative level* (from *Signature Characteristics Module*) that can be filled with a number that goes from 0 to 100, which implies that this information can be subjective, because different persons can give different values to this field.

- The *Data Link Protocols Module* was not used in a single signature. In a first view, can be assumed that this module is useless, but if an attack uses this kind of information (a MAC spoof, for example), the model already support it.

What can be said is that the model supplies the need for a standard to store attacks signatures, keeping the exchange of these information very simple and efficient.

6 Conclusion

The model was submitted to a test, in which different kinds of attacks were codified. The test was successfull, because every attempt to codify an attack could be done.

The model allows the exporting and importing of important information about attack signatures, between different IDS, in a simple way through the use of XML specifications. This feature is very important, because the more information is known by an IDS, the more rules can be built to make it more efficient.

References:

[1] Cansian, Adriano M.; Souza, Marcelo de; Silva, Artur R. A. - Developing an Attack Signature Standard – *In Proceedings of SAM'02 - The 2002 International Conference on Security and Management*

[2] 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)

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

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

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

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

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

[8] 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.