Intrusion Detection and Internet Services Failure Reporting System

Przemysław Nowak, Bartosz Sakowicz, George Anders, Andrzej Napieralski
Department of Microelectronics and Computer Science
{pnowak ,sakowicz ,napier}@dmcs.pl, george.anders@attglobal.com

Abstract

In this article the Intrusion Detection and Internet Services Failure Reporting System created to detect unwonted manipulations to computer systems, mainly through the Internet and checking all active services in secure network is described. The application was written using Java Enterprise Edition (JEE) technology and jpcap library,so the application can work on many different operating systems. Intrusion Detection System is based on signatures method detection.

1. Introduction

When securing a network, administrators have to use many different tools. For each operating system different applications have to be used, regardless they are doing exactly the same. Although functionality of them is similar, administrators have to spend a considerable amount of time to read a documentation and learn how to use a new tool. To minimize this effort a specialized tool securing network and checking available services is needed [7]. In this article an universal system for monitoring network traffic and services available on different operating systems (with Java Virtual Machine installed) is described. The application is a passive system, so problems are only reported. It can be changed, but the outgoing module should be rewritten first. An Intrusion Detection System (IDS) is a defense system, which task is to detect hostile activities in a network. The key is then to detect and possibly prevent activities that may compromise system security, or a hacking attempt in progress including reconnaissance/data collection phases that involve for example, port scans. To recognize possible attacks systems are examined for any abnormal behavior. This may be helpful in detecting real attacks.

2. Application structure

The described system integrates the test Intrusion Detection System with the Monitoring Failure System. Up to now these tools were different Systems with different configuration methods. This system ensure one tool to detect failure services and intruders.

To make application available for many operating systems authors decided to use Java language (Java Virtual Machine can be installed on most popular operating systems). To get a low level access to the ethernet network devices a jpcap library was used. The jpcap get an access to the standard operating system function written in C and C++ language by Java Native Interface (JNI).

The JNI is a standard programming interface for writing Java native methods embedding the Java virtual machine into native applications. That allows Java code running in the Java Virtual Machine (JVM) to call and be called by native applications (programs specific for hardware and operating system platform) and libraries written in other languages, such as C, C++ and assembler.

2nd International Conference on Dependability of Computer Systems(DepCoS-RELCOMEX'07)
0-7695-2850-3/07 $20.00 © 2007 IEEE

The main aim of the system is to detect all types of malicious network traffics and computer usages. The intruder detection and the reporting failure help administrator teams to keep the computer system safe. The second part of the system checks all services in the network and reports if one or more services work inappropriately. The Application consists of two main parts which are available to user [1]:

- Network Intrusion Detection System (NIDS)
- Internet Services Failure Reporting System (ISFRS)

These two parts can work independently. The application was tested on various operating systems, inluding Linux Debian, OpenBSD and Microsoft Windows XP Professional. The application has a modular structure, every part of the system consist of cooperating modules.

2.1 Intrusion Detection System

The most important activity of the system is an intruder detection. IDS monitor packets on a network and compare them with a database of signatures or attributes for known malicious threats. This is similar to the way in which most antivirus systems detects viruses [5]. The important problem is that there is a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to the signatures database. To keep the application up to date, authors used the signatures database from project Snort [4].

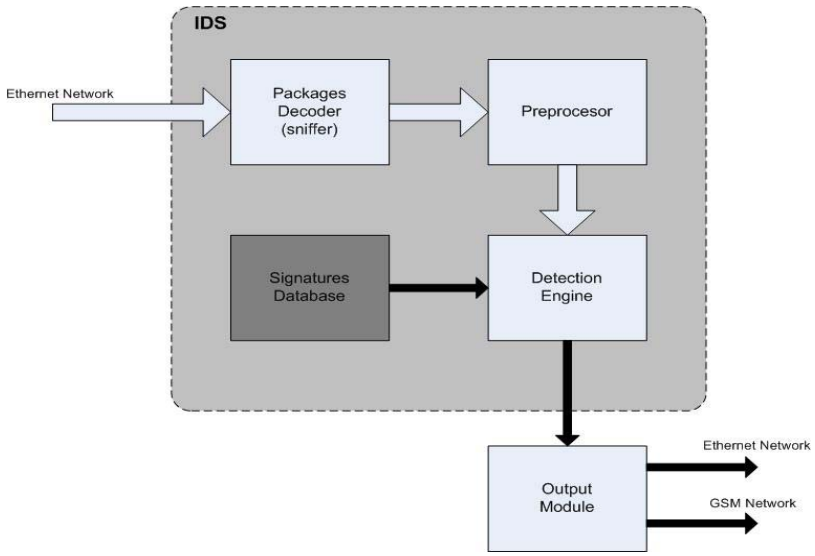

Fig.1 Intrusion Detection System modules.

IDS contains four modules (Fig. 1):

- Sniffer getting packages from a network,
- Preprocessor preparing packages for analysis,
- Signatures databases – database of patterns of known malicious threats,
- Detection engine – compares packages with signatures.

Sniffer – a packages decoder specific for every operating system, usually written in C or C++ language. Because a part of JEE is JNI, classes written in Java language can be integrated with C and C++ (Fig. 2).

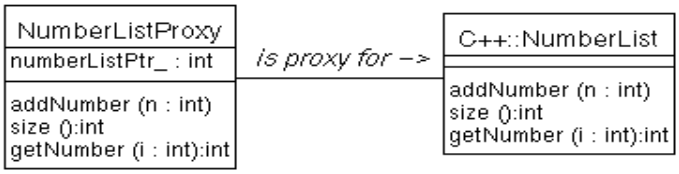

Fig. 2 Java and C++ integration example [6]

To cooperate with different OS the IDS should have implemented an universal method to switch Ethernet card to promiscuous mode. The jpcap library is responsible for this task and requires libcap or wincap library available on the operating system.

Preprocessor is responsible for preparing packages for analysis. Sometimes processes can cause some problems with detection. To effort this preprocessor analyzes connection and prepare package sequence for engine module.

Signatures database – specially prepared patterns database. Every incident is analyzed to get a regular expression describing the type of attack attempt.
There are a lot of signatures in the database. Authors of this project used signatures form the project Snort, because the database is still being developed by the Snort Project Team, so updates are often released (Fig. 3) [4].

Detection Engine takes packages from preprocessor and compare them with special signatures from the database. Result of the compassion is sent to outgoing module, where a report is prepared.

```
alert tcp ![192.168.1.0/24,10.1.1.0/24] any -> \
[192.168.1.0/24,10.1.1.0/24] 111 (content \
"|00 01 86 a5|";    msg: "external mountd access";)
```

Fig. 3 Rule example[4]

2.2 Internet Services Failure Reporting System

The second element of the system represents application monitors services available in a secured network (HTTP, SMTP, POP3 etc.). In some time intervals
(defined in a configuration file) every service on the server is checked and a report is generated. If any service is down, a short report is sent to administrators. This element has two basic modules and a few small plugins for checking each of services (Fig. 4).
Plugins can be written by any programmer who knows the basics of Java language.
Only one condition has to be respected: a plugin has to implement ServiceTest interface (designed to communicate between a plugin and a system) as shown in Fig. 5. This interface was designed specially to get more functional testing module. The basic function *check()* gets a simple answer: "Service is working or not". Because services not always work on a default port number, the function gets a possibility to define a port number for each service.
The *getMessage()* function can be used to return a more detailed information about the

service and allows administrators to decide how many information about the service they want to receive.

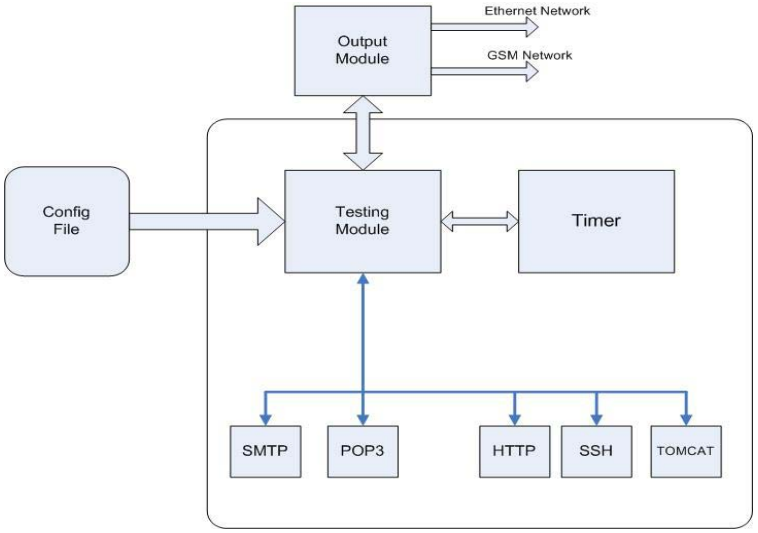

Fig. 4 Internet Services Failure Reporting System

```
public interface ServiceTest{

	boolean check(String hostname);
	boolean check(String hostname, int port);
	String getMessage();
}
```

Fig. 5 ServiceTest interface

In the current version of the system, following test modules: dns, telnet, ftp, smtp , ssh, pop3,spop3, imap, simap, http, https, tomcat are avaliable. An example of smtp test module has been introduced in Fig. 6. The presented example shows a testing module checking availability of the smtp service. In this case administrators get a short information about working status of the outgoing e-mail system. The complex module can check technical details of the service (authorization, availability of the single page), and return status messages.

```
// imports ommitted
public class SmtpTest implements ServiceTest {
	private String message = null;
	public boolean check(String hostname) {
		return check(hostname,25);
	}
	public boolean check(String hostname, int port) {
		String tmp=null;
		Socket client=null;
		BufferedReader in=null;
		StringBuffer rep=new  StringBuffer(1);
		try{
			client = new Socket (hostname, port);
			in = new BufferedReader (new InputStreamReader
			(client.getInputStream()));
			tmp=in.readLine();
			message= "SMTP is working";
		} catch(ConnectException e){
```

```
                    message= "SMTP does't work: Connection Refused";
                    return false;
              } catch(IOException e) { e.printStackTrace();
              } finally {
                    try{
                          if(client!=null) client.close();
                    } catch(IOException e) { e.printStackTrace() }
              }
              return true;
        }
        public String getMessage() {
              if(message!=null) return message;
              return "Host wasn't check!!!"
}}
```

Fig. 6 ServiceTest class example

There is one more module used by both parts of the system. It is the outgoing module sending reports to the network administrators team. As mentioned before, the application was designed to be independent from the operating system.
The output module contains a simple SMTP server to send reports by emails and a GSM module to send SMS messages with short report. The reporting module was split into two submodules to ensure that administrators will receive a notification immediately.

To send SMSes, the system uses public GSM gateways but it is possible to use a dedicated GSM server card, which is, of course, more reliable.

The system configuration is very simple. A special file contains in each line address of server and services to be checked. An exemplary line is presented below:

```
mail.dmcs.pl: smtp,pop3,imap,dns,imaps,pop3s,ssh
```

3. System requirements

The Java technology allows for multi-platform usage of the described application, only the Java Virtual Machine and libcap or wincap libraries are necessary for the system to work properly. The signatures database can be downloaded from web page of the Snort project. System does not require a powerful hardware, the application was successfully tested on the Intel Pentium III 1GHz processor with 512MB operating memory.

4. Further research

The lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to signatures database is a major defect of the signatures checking method. To prepare the Intrusion Detection System to detect new types of attacks (without signatures) authors are currently developing the new module detecting attack attempt with usage of heuristic algorithms [9].

5. Conclusions

The main goal of this project was to create an application able to work on most popular network operating systems and containing the functionality of a few different tools usually used by network administrators. There are a few applications on the market, which fulfill these requirements, but are expensive and inappropriate for authors' needs, therefore it was decided to begin a research.

Described project received distinction from the Association of Polish Electrical Engineers (SEP) December 2006.

Acknowledgements

This research was supported by the Technical University of Lodz Grant K-25/1/2006/Dz.St.

References

[1] P. Nowak, B. Sakowicz, A. Napieralski. "System wykrywania włamań i powiadamiania o awariach serwisów internetowych", Mikroelektronika i Informatyka, Łódź 2005, pp. 33-36, ISBN 83-922632-0-0

[2] P. Nowak, "System wykrywania włamań i informowania o awariach serwisów internetowych", Master Thesis, Technical University of Lodz, July 2006.

[3] R. G. Byrnes D. J. Barrett, R. E. Silverman, "Linux.Bezpieczeństwo. Receptury.", O'Reilly, 2003.

[4] B. Caswell, J. Hewlett, "Snort users manual", 2003.

[5] E. Amoroso. Sieci: Wykrywanie intruzów. Wydawnictwo RM, 1998.

[6] B. Foote, "Integrating Java with C++", JavaWorld.com, 1996

[7] M. Wójtowski, B. Sakowicz, P. Mazur, "Kompleksowy system o wysokiej dostępności", Mikroelektronika i Informatyka, Łódź 2005, pp. 211-216, ISBN 83-922632-0-0

[8] B. Sakowicz, J. Wojciechowski , K. Dura., "Metody budowania wielowarstwowych aplikacji lokalnych i rozproszonych w oparciu o technologię Java 2 Enterprise Editon", Mikroelektronika i Informatyka, maj 2004, KTMiI P.Ł. , pp. 163-168, ISBN 83-919289-5-0

[9] Masahiro Yamauchi, Thsimasa, "A Heuristic Algorithm FSD for the Legal Firing Sequence Problem of Petri Ncts", Hiroshima University.