Proceedings of the 5th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, Madrid, Spain, February 15-17, 2006 (pp166-171)

Implementation of authentication techniques across closed ports.

RUANO, ILDEFONSO; GALAN, SERGIO; GARCIA, SEBASTIAN; VICIANA, RAQUEL
Electronic, Telecommunication and Automatic Engineering Department
University of Jaen
E.P.S. de Linares, C/ Alfonso X El Sabio 28, 23700, Linares (Jaén)
SPAIN

Abstract: This paper describes several authentication methods and presents a software packet that implements them in order to improve the application security of programs that offers services in Internet. The communication in the server side is done at the link layer. So it is possible to transmit data although the ports are closed. At advantage, these services can not be detected when the attackers use portscans to probe networks and determine what services are active in the host because the ports are closed at transport layer. Four different methods have been developed. These techniques can be used in different environments and situations in order to obtain several security levels. A server and a client program which implement the four methods have been created. If the authentication is positive the server offers the possibility to the client of running some commands as a request of the client. The client helps to obtain the authentication process and makes it easier.

Key-Words: Internet Security Authentication TCP Ports Link-layer Cyber-defence Portknocking

1 Introduction

When an application offers a service in an IP network the security is one of the most important issues that the developer must consider. This security lies in two main aspects, the first one is the application security and the second one is the application-access network security. The application security is discredited daily due to multiple alerts that appear everyday. Therefore an optimal security level only based on a well-done application is an impossible idea. The applications must be helped with software and networks devices that minimize the attacks. This work describes a series of techniques that add greater security to Internet services and are based on the ports hiding process at transport layer. In addition it is presented an application that implements these techniques and shows its possibilities.

The first step to attack network software is to detect its presence. For this reason attackers usually employ portscans during reconnaissance to probe networks and determine what services are active on which hosts [1]. For the purpose of providing services to unknown callers, a service contact port is defined. A list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port". Therefore if an attacker obtains the list of the open ports it is equivalent to obtain a list of open network services. As a next step, they obtain a great amount of information about the service and then they can launch frequently pre-scripted attacks against well-known vulnerabilities (bugs). The developed methods shown in this work avoid attackers from reaching this phase and present these features:

a) They are situated as a protection layer above the network services.
b) They do not show any open port at transport layer.
c) They obtain the input data from a network interface (at link layer).

In order to facilitate the data reading, the libpcap library has been used [2][3]. Lipcap is a system-independent interface that provides a framework for low-level network monitoring. This software is used by several applications (tcpdump, arpwatch) and other related works [4][5].

The received data are filtered and analyzed. Then if this data are correct, arrive in the correct order and on time then the authentication process is positive. In this case the client can run a script in the server which can open/close services, perform other system management functions or configure a firewall in order to open/close ports.

The implemented methods are different adaptations of portknocking technique. The most common implementation of portknocking consists in reading the firewall log file, looking for a combination on successive connection attempts to different ports [6].

This procedure constitutes an additional layer of authentication that mainly provides two benefits:

1. It hides non-public network services to non-authorized users.
2. It avoids DoS (Denial-of-Service) attacks. Firstly a lightweight process, which can discard practically all

the access attempts, is placed before another one which can perform many heavyweight cryptography calculations and exhausts server CPU resources [7]. Secondly it protects from SYN flood attacks (it sends TCP connections requests faster than a machine can process them).

2 Portknocking Methods

Next some of the most common methods that are able to obtain authentication across closed ports are described. In all these cases it is supposed that exists a client C which wants to obtain an authorization from a server S which does not have any open port at transport layer and is listening at the link layer.

Several implementations that obtain the authentication based on portknocking can be located at the address http://www.portknocking.org.

2.1 Using header fields of the protocols

Protocols TCP/IP headers contain fields that have a constant value for the entire route from the origin to the endpoint. This can be used to perform an authentication process that works as it is explained bellow:

C sends a prearrange SYN packets sequence. The SYN packets are connection request packet and each packet has a different port number. These requests are known as knocks, therefore, and so these techniques are known as portknocking. S receives and processes these packets at the link layer and compares the ports sequence received with a fixed configured sequence. If both of them are the same sequence then C is authenticated by S and can order the running of a command script in S. In order to use this method, some questions have to be taken into account:

1. The random success probability caused by a brute force attack is very small: $1/(2^{16})^N$, where N is the number of knocks of the sequence and 2^{16} is the number of ports that exists in the transport layer.
2. If someone listen the data traffic it can easily perform an attack by an imitation process.
3. The IP packet order can not be guaranteed in Internet owing to the IP protocol features. Therefore the value of N must not be very high and the maximum delay interval between knocks must not be very low. If these points are not observed there is a probability of putting the packets out of order.
4. If there is a firewall between C and S (in the route) is quite feasible that the firewall discards the SYN packets (Firewall usually discards practically all the ports connection attempts). In this case S does not receive any SYN packet.

Most of these points must be taken into account for the next methods.

2.2 Sending encrypted information as data

C shares a key with S; C uses this shared key and sends a packet with encrypted information to S. S waits for the arrival of a packet that (i) obeys a condition (i.e. port number) and (ii) is encrypted with the shared key. When the packet arrives S runs the configured action which can be to open a specific port for a time and then S waits for a second SYN connection request packet. If the time expires and does not arrive any packet, S closes the service.

2.3 Modified SYN packets

This method is similar to the last one, the main difference consist in the beginning: C sends to S only one SYN packet that contains the encrypted key and the port number that C wants to open. If the key is correct S open the port and makes a three ways TCP authentication resumes, otherwise S rejects the connection.

2.4 Sending encrypted data prior to knocks sequence

This method does not use a prearrange SYN packets sequence. C creates a random port sequence, C encrypts this sequence with a S-shared key and sends it to S in a packet. S receives the packet, obtains the port sequence and waits for the packets series that matches with the port sequence. If C sends the packets correctly, S can authenticate C and the process finishes.

2.5 Knocks of encrypted data

Portknocking original implementation obtains the knock information from the “Iptables” firewall log files (linux). This implementation opens/closes ports configuring the firewall. The system security improves if the knocks information is encrypted with a shared key.

2.6 The dark side

Although these techniques have been developed to increase the security level of a system, they can also be used in a malicious sense [8]. Nowadays hackers use portknocking techniques to hide malicious software (malware). They install it in computers without authorization (trojan or backdoors). When these methods are being used its detection becomes difficult. However, the software presented in this work requires super user permissions for its installation and it is visible like any other process. This circumstance avoids the installation without permission of the system administrator.

3 Software Structure

A server program has been developed (PuerTockd) which implements different portknocking methods. These methods are explained in detail in this section. For the implementation of the server program open source libraries have been used and the final program itself is too opensource (It can be downloaded from http://www4.ujaen.es/~alonso/pk.htm).

The developed software general structure is shown in figure 1.

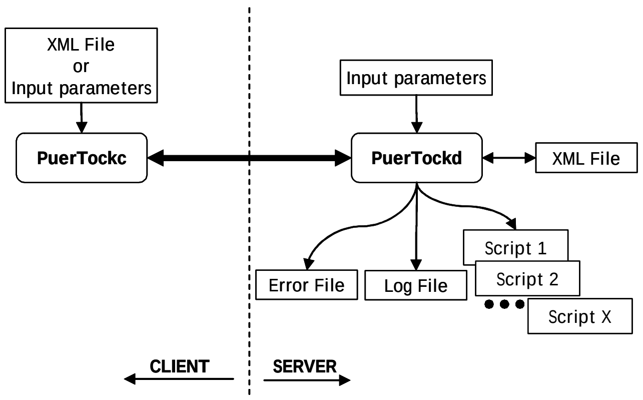

Fig. 1: developed software general structure.

The most remarkable features of PuerTockd are the following ones:

- It has been written in C language and it works in GNU/Linux environments using the “Libpcap” library.
- It implements four authentication models through closed-ports.
- The configuration can be carried out of two different forms: by means of an external file in XML format and by means of commands. In order to work with a XML file the “libxml” library v.2. has been used.
- It is designed in form of a daemon to be run in background mode. Once in execution, it analyzes all the incoming traffic of the link layer applying filters that allow detecting a valid sequence for the implemented method. It works with MAC protocols, Ethernet and Wi-fi.
- If a positive authentication takes place, the server allows running different commands scripts (Bash shell) which implement some of the possible tasks: remote command execution, the control of a Firewall, or open and close services.

3.1 Internal Estructure

Daemon PuerTockd has the internal structure shown in Figure 2. This figure shows the three main blocks:

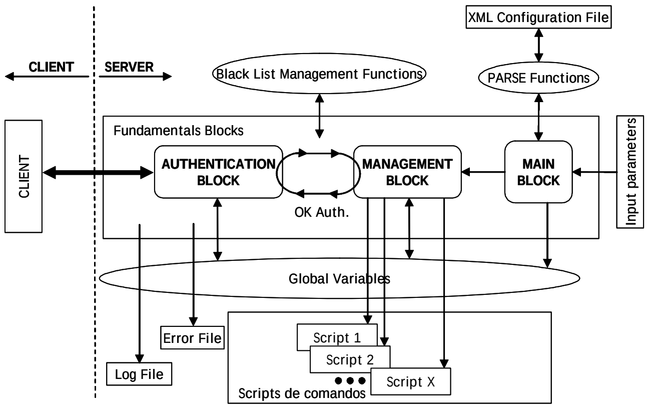

Fig. 2: PuerTockd internal structure.

Main Block: This block analyzes the command line and the configuration file, also turns the process to background mode and calls the corresponding authentication function.

Management Block: it manages the correct authentications and calls the corresponding script.

Authentication Block: it implements the corresponding authentication techniques.

3.2 Server Configuration

The basic server operation can be determined by one of the two previously commented methods. Figure 3 shows the XML file structure using a tree representation.

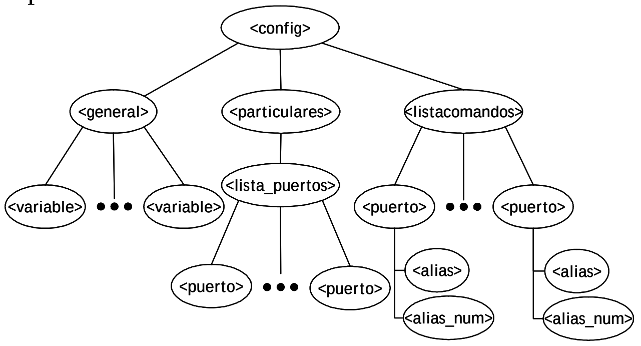

Fig. 3: Configuration file structure.

The most important options that can be configured are:

- The authentication method to implement.
- The configuration file and its location.
- The messages file of events registry and its

location.
- The error messages file and its location.
- The client IP address. The authentication requests from different IP addresses will be ignored.
- The password to use in the methods that require it.
- Execution in background.
- Ports Sequence to validate.
- Input Interface.
- Maximum delay interval between knocks.
- Period of time in which a not authenticated direction IP remains in the address black list.
- Different command sequences that the client can run once authenticated.

3.3 PuerTockc Client

In addition, a client has been implemented via a program (PuerTockc) and is developed also in language C. This program allows testing the different authentication methods and works under GNU/Linux. The use of this program is not necessary in all the methods, since some of them can be proved with standard software of other S.O.

Like PuerTockd, PuerTockc can be configured through the input parameters or via a XML file.

4 Implemented Methods

The four authentication models are adapted to different environments and security requirements.

4.1 "knockmal" Method

This method is presented in section 2.1. The client application, in order to be authenticated, sends a request connection packets sequence (SYN). Each of these packets includes the port number in the corresponding field of the TCP header. These requests are known like knocks, for that reason this technique is also known like portknocking. PuerTockd receives and processes these packets in the link layer, comparing the sequence of ports received containing the formed sequence. If the received sequence is the correct sequence the authentication takes place, and the client is able to run the desired actions in the server. Normally this method can not be used when the server is protected by means of a Firewall, because this one can block the packets sent to non-allowed ports, and therefore the packet sequence is not validated by PuerTockd.

Figure 4 shows a scheme-example describing the operation of this method.

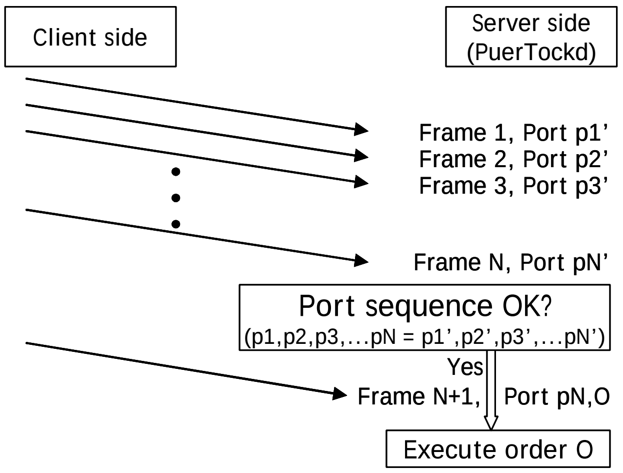

Fig. 4: knockmal method Scheme-example.

4.2 "dns_auth" Method

This method solves the problem that can be produced with the use of the previous method, because it only uses a fixed port to make the authentication that can be programmed in the Firewall. This method uses a DNS protocol to send only one packet [9]. The client application, in order to be authenticated, communicates with the PuerTockd server by means of DNS message. The authentication message is inserted in a domain question field of the DNS protocol. Figure 5 shows the DNS message that is sent if the password is "miclave" and the order "ftpopen" (to open the FTP service in the server).

In addition, figure 5 also shows the typical scene of use of this method.

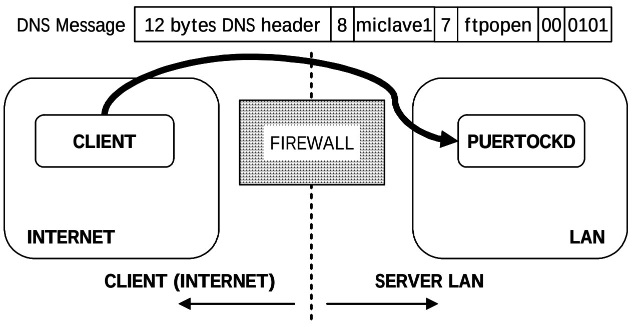

Fig. 5: Typical scene of use of "dns_auth".

In order to make an authentication, commands as "dig" (linux) or "nslookup" (windows/linux) can be used instead the PuerTockc client. As the password is transmitted in plane text, the security level is not very high. Nevertheless, this method can be very useful due to its easiness of use in environments like the one described above.

4.3 "knockudp" Method

"Knockudp" method implements encrypting techniques. Therefore, it is better adapted than both previous methods in non secure environment.

This method begins transmitting a set of messages UDP from the client application (PuerTockc) that include a certain ports sequence. This sequence must be verified by the server (PuerTockd).

When the sequence is correct, PuerTockd uses the IP address and the port used in the last packet of the received sequence to transmit to PuerTockc a UDP message. This message includes a random vector of size M. Then, PuerTockc must use a both client and server well-known private key to codify the received vector and to transmit it to PuerTockd in a new UDP message. When PuerTockd receives this UDP message and verifies the content, the client is authenticated and is able to run the desired actions in the server.

By default, the coding function used is RIPEMD, despite it is also possible to use others like MD5 or SHA1. Figure 6 shows the typical interchange of messages that takes place in the implemented method.

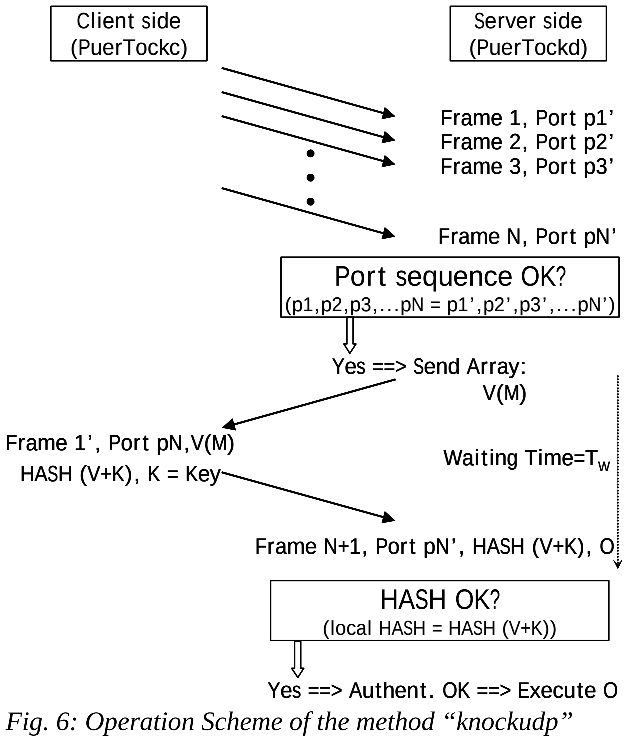

Fig. 6: Operation Scheme of the method "knockudp"

4.4 "knockcifrado" Method

The essential difference of this method referred to "knockudp", is that transmissions from the PuerTockd server to the client are not required.

The first stage of this method is similar to the previous method. Nevertheless in this one the last packet sent from the PuerTockc client to the PuerTockd server includes encrypted information, a time-mark and a common key (allows the server to determine if the client can be authenticated). Figure 7 shows an operation example of this method.

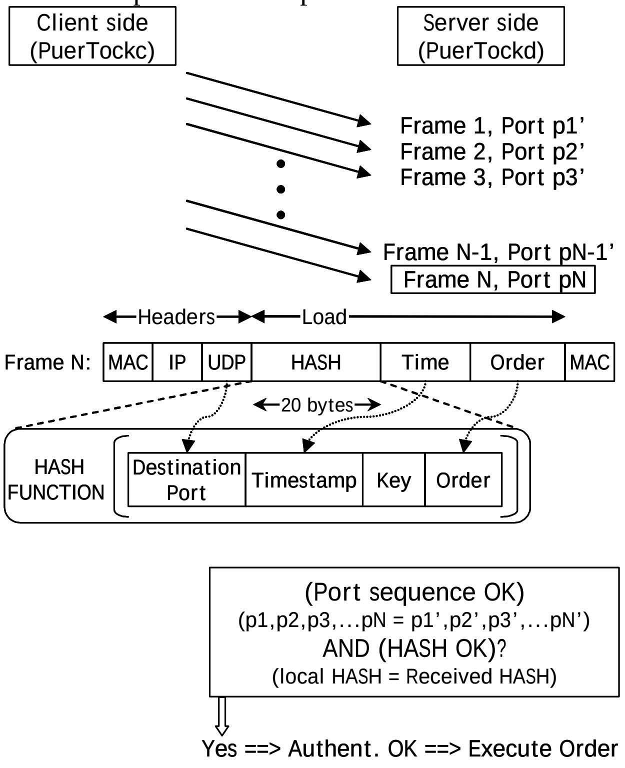

Fig. 7: Operation Scheme of the method "knockcifrado"

5 Other system features

In this section we describe some relevant features added to the system that were merely cited in the general software structure presented in section 3. These features are the events log, the IP address black list and the command scripts.

5.1 Events Log

The server system events log allows controlling the more relevant events that take place in PuerTockd as: wrong access attempts, correct access attempts, mistakes, etc. The output messages are formatted using XML syntax. The error messages (ERR by default) and the rest of messages can be defined in different output files. This is equivalent to the standard outputs "stdout" and "stderr".

5.2 Black List

The software oriented to provide security in the applications must be ready to tackle the committed attacks produced against it. One of these attacks is the attempt to discover the codes (or the port sequence) via a brute force mechanism. Another problem is the possibility of an attacker making multiples attacks as authentication attempts, mainly when these attempts have the goal of diminishing the server process capacity or complicating the regular customer's access.

Even though PuerTockd adds a minimum load to the system, it provides a set of functions that deal with the creation and maintenance of a malicious machines IP address list. So, when an IP address exceeds a certain threshold of connexion attempts is included in the black list and its traffic is discarded (pcap library offers this possibility via filters). After a configurable period of time the IP addresses loaded in the black list are removed. Furthermore we can configure the next parameters: numbers of wrong attempts to be included in the black list, waiting time after an IP address in the black list is marked to be deleted, etc.

5.3 Scripts

Once the client is authenticated, the client can run in the server certain commands (orders) that have to be previously configured. Furthermore, the commands already selected to be run in the server can be modified and new ones can be added using the PuertTockd's configuration file.

The commands or scripts configured by default can be grouped in three sets:

1. Open/Close services: sshd, proftp or any others that are placed in the directory /etc/init. (These services are only accessible for the IP that reached the authentication).
2. Firewall Iptables control, in case the firewall is running in the same host that PuerTockd.
3. No critical management functions: as an example a management function that allows performing a mysql data base security copy of the server has been included.

6 Conclusion

The implemented methods provide the capacity of adding a higher level of security to Internet servers. So, the developed software hides the server at the level of the transport layer but allowing access from the software of an easy client. The software shown in this paper offers a security level that is not enough for a normal server. Nevertheless this implementation can add a new obstacle to the hackers who want to access maliciously to the system. It is necessary to use it integrated with other security technologies in order to robustly protect a server.

There is a Windows version of libpcap that is called winpcap [10]. Winpcap has been used by other investigators and as future work we consider the developing of a new Windows version as well as the implementation of new Portknocking methods [11][12].

References:

[1] Atighetchi, M.; Pal, P.; Webber, F.; Jones, C.; Adaptive use of network-centric mechanisms in cyber-defence, *6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2003, pp.183-192

[2] Carstens T.; Programming with pcap, http://www.tcpdump.org/pcap.htm

[3] NAU's Computer Systems Engineering; Packet Capture with libpcap and other low level network tricks, http://www.cse.nau.edu/mc8/Socket/Tutorials/

[4] Chen, R.; Gao, J.; Hua, C.; Higen: An intelligent system for misuse detection *Proceeding of 2004 International Conference on Machine Learning and Cybernetics*, 2004, pp.2775-2778

[5] Corley, M.W.; Weir, M.W.; Nelson, K.; Karam, A.J.; Simplified Protocol Capture (SIMPCAP), *Proceeding 5th annual IEEE SMC Information Assurance Workshop*, 2004, pp.176-182

[6] Kzywinski, M.; Portknocking: Network authentication across closed ports, *SysAdmin Magazine*, No.12, 2003, pp.12-17

[7] Barham, P. et al; Techniques for lightweight concealment and authentication in IP networks, *Intel Research Berkeley* 2002, http://www.intel-research.net/Publications/

[8] Martin, K.; Click on this, you muthas, The Register, http://www.theregister.co.uk, 2004

[9] Mockapetris, P.; RFC1035, Domain Names Implementation and Specification, STD13, *USC/Information Sciences Institute*, 1987

[10] Risso, F.; Degioanni, L.; An architecture for high performance network analysis, *Proceedings 6th IEEE Symposium on Computers and Communications*, 2001, pp.686-693

[11] Degioanni, L.; Baldi, M.; Risso, F.; Varenni, G.; Profiling and optimization of software-based network-analysis applications, *Proceedings 15th Symposium on Computer Architecture and High Performance Computing*, 2003, pp.226-234

[12] Zhimin, W.; Xiaolin, J.; Restoration and audit of Internet e-mail based on TCP stream reassembling, *Proceedings ICCT International Conference on Communication Technology*, vol.1, 2003, pp.368-371