Global Information Assurance Certification Paper

Forgetting to lock the back door.
A break-in analysis on a Red Hat Linux 6.2 machine.

Gary Belshaw

August 04, 2002

GSEC v1.4 Practical Paper (option 2)

Abstract

This document is intended to highlight the steps taken in ascertaining the level of damage done in a network break-in (or hack attack) on our system, and the steps taken in rectifying the damage. Using the crisis case I encountered in a small company, I will demonstrate how to gather the evidence, secure the network, and provide suggestions for amendments to the existing system to minimise the chances of a repeat break in.

I will also discuss some of the tools and utilities I used in order to perform this task and give comments to how they should be applied.

Overview

These days, many small companies have Internet access, and the majority of these companies rely for this access on systems which have either been incorrectly set up, or have had little or no maintenance performed on them since initial installation. This is usually due to the failure of the company to dedicate a full-time administrator or employ a third party to do system administration work. Reasons for this include financial constraints, or an unwillingness to recognize the dangers that they are facing. (The “ostrich with its head in the sand” approach)

Many companies I have been in contact with do not fully understand the importance of the data contained on their systems, and the majority of them are extremely disturbed when they find out that the value of their data is invariably far higher than the total cost of the hardware on which it resides.

When one takes into account the data loss, the time spent in recovering from a break-in and the downtime as a result of action being taken to reinstall operating systems and reinstate data, a single break-in can have a great effect on the future of the company.

Add to that, the possibility of company confidential files and personal data being stolen and misused by third parties, and you will begin to see the impact that network security (or the lack of it) has on people’s lives.

The case highlighted in this document describes a RedHat 6.2 machine that was compromised via the Internet, and used without the owners’ knowledge to perform scans for vulnerable servers, and report those results to the intruder.

It also shows the losses incurred by the company due to the downtime that occurred as a result of the break-in, and the knock-on effects that this had with the users of the network.

The Setup

The company mentioned in this document had a small Windows 98/Windows 2000 network, which was connected through a Default install Red Hat 6.2 machine, acting as a gateway/firewall on a 256k leased line. The gateway machine was set up so that no **Inetd** functions were running, and the only administration that could be done on the machine was through the local system console (i.e. the keyboard!). The internal network was a 192.168.x.x. Network which was using Network Address Translation on the firewall to minimise the number of fixed IP addresses being used on the outside of the firewall. This also helped to ensure a degree of security by masking internal IP addresses from outside view.

The Gateway was running **Bind** in **forward-only** mode to provide DNS services to the internal network. In addition to this, there was an e-mail server (FreeBSD 4.2) and a Web server (FreeBSD 4.2) located outside of the firewall area of control.

I joined the company a few months after the system had initially been commissioned, and it seemed to be running nicely. However, after a few weeks, users started to notice degradation in network throughput, this was especially acute when using the Internet. In addition to this, a few small problems (like machines locking up intermittently) arose from time to time with some users.

These problems were rectified, and were usually attributed to Windows problems. In nearly every case, a reboot fixed the problem. Therefore we thought nothing more about it.

Discovery

One morning, I received an e-mail forwarded by our upstream provider from the administrator of a network in Canada, asking why a machine using one of our IP addresses had been scanning their network.

The IP address in question belonged to our gateway/firewall box, so I immediately went over to the unit and tried to login, however, the machine would not accept my password - We had been compromised!

Collecting the evidence safely

I disconnected the gateway machine from the router, (much to the distress of everyone in the office, as they lost all Internet access) powered the machine off, and removed the hard disk…

Generally, it is not a good idea to reboot the compromised machine if you need to collect evidence of a break in. Once he is in, the intruder can set up all manner of logic bombs and booby traps, these activate automatically on boot-up, destroying all traces of the break-in.

The recommended way is to try to capture a "live" snapshot of all the partitions on the hard disk (including the swap partition), and if possible, get a memory dump.

Unfortunately, I did not have this option, as I had been locked out of my machine.

So the next-best way to access a compromised machine is by booting from another hard disk, or if this is not possible, a floppy, or bootable CD-ROM. Then you can mount the required partitions from the compromised Hard Disk. This procedure ensures that the swap file is left intact and unaltered from the time the compromised machine is powered off, and no Trojans or booby traps are executed at startup.

Please note, **powering down a compromised machine should be done by switching it off at the mains**. **Do not put the machine through a normal shutdown procedure**. In performing a normal power-down procedure, any Trojans executed by shutdown scripts will run automatically, and you could lose everything you want to capture.

Linux does not like being powered down this way. (Plus there is a chance of trashing your hard disk) However, I have never had a hard disk crash from doing this (and I have done it a few times). This is not to say I recommend doing this; it is simply a case of a choice between the devil and the deep blue sea, and in my case, I had no other option.

After installing the hard disk from the compromised machine into another Red Hat machine, (a standalone unit with no network access) and then mounting the partitions manually. I then copied the entire contents of the disk to two separate hard disks, then powered the machine down and removed the original compromised disk in order to keep it as evidence should it be required. I also removed one of the copied hard disks so I could keep it as a master copy, should I need to copy it again it in the future.

To copy the compromised disk, I used **dd**. This is a good tool for collecting forensic evidence, as it supports block devices, (such as hard disks, floppies

and Tapes) and it is free. There are other third-party tools available to do this as well, but they are usually expensive. **dd** is available on your system from any standard installation, and is a good option.

Examining the evidence

Now I had a copy of the compromised hard disk, and I could safely start to look for evidence without fear of damaging the original.

The first thing that I did was to look in **/var/log** for any log files, which would inform me of any actions that the intruder had performed. As expected, these had been deleted to aid him in covering his tracks.

I then looked at **/root/.bash_history** and found that the attacker had missed deleting the contents of this file. Further examination of .bash_history revealed that my intruder had downloaded and installed a root kit called **t0rn** from ftp.tomb.org.

On deeper inspection of the hard disk copy, I also found files located in **/dev/caca /dev/dsx** and **/dev/zzy.** Some of these files were part of the t0rn rootkit, but there were others that had no association with t0rn.

Some of these were configuration files for Trojanised versions of ps, ls and netstat. There were also some executables, including **Linsniffer**, an application used to sniff TCP/IP packets from a network and capture those packets to a log file.

There were also extensive TCP/IP logs, containing user names, passwords, e-mails and even some credit card data, which had been sent via a clear-text email.

After doing a search for the names of some of the executables, it quickly became clear that I had multiple root kits (or parts thereof) installed on the machine, and I could not rule out the possibility of our intruder having done more damage to other machines on our network. The additional root kits that were present were **LRK4 and Lamerk**

The **Lamerk** root kit seemed initially to have been the likely culprit for the outbound scans on other networks, plus further investigation unearthed log files of scans which had been done and IP addresses of machines that were vulnerable. It also seemed likely that my machine had been initially compromised by this very root kit, and was now propagating the same root kit to other machines on additional networks. However, the creation of log files is unusual for **Lamerk**, as there is a bug in the source code, which prevents the automatic writing of logs by the scanner included in the kit. This led me to suspect that another scanner was responsible for the logs I had found.

In **/dev/zzy** I found an executable file, and some configuration files which indicated that my machine was connecting via an Internet Relay Chat client to channel **#Bulgaria** on **irc.tomb.org** and that there was some sort of remotely accessible back-door available through this client, (an egg-drop bot) which would allow IRC users with the appropriate credentials to connect to my machine as root. This seemed like a more likely way than **Lamerk** that could be used to run intrusive network scans on other hosts at will by these users.

At this point, I had to decide whether to continue with my forensic investigation, or to rebuild the gateway machine, and restore the company's Internet access.

The company won…

Fixing the damage

Unfortunately, this was not a simple matter of just reinstalling the operating system on the gateway machine. The intruders had had plenty of opportunities to acquaint themselves with the other machines on my network. I had to make sure that I was not just closing the front door, in order to find out that a back door was wide open.

Luckily, the two FreeBSD machines (the Web server and e-mail server) were not compromised. A **Tripwire** scan, running **Chkrootkit**, and doing an **Nmap** scan of them revealed no suspicious ports open or root kit files confirming this.

Tripwire generates an MD5 hash database of all critical files on your system.

You can then scan your system to check against that database and find out whether your machine has been compromised. Initially, Tripwire checks all default system files, with extra configuration, you can cover any configuration you decide to implement.

Save this database to a floppy or a secure remote location, and scan regularly against this database to be sure your executable files are what they claim to be. **Chkrootkit** is a great little utility that scans your machine for evidence of a root kit. It does not use any of the existing binaries on your system, and easily fits on a floppy disk.

Faced with the sniffed password evidence I retrieved from the gateway machine, I decided to change every user's email password in order to minimise any chance of an attack through a compromised user account on my e-mail server.

To make this task slightly easier, I decided to use **APG** in order to generate six-character complex passwords for all users of the e-mail server. I then wrote down all the individual passwords and distributed them to the respective account-holders, advising them to change their passwords as soon as possible,

and from then on, on a regular basis.

APG is an automated password generator, which will generate passwords according to rule sets that you define.

A short aside about password security…

In a perfect world everyone would have minimum 10 character-length passwords that are a random mix of numbers and upper and lower-case letters (with a few symbols thrown in for good measure).

However, this is not a perfect world (sadly), and users still insist on using "password" or "letmein" as passwords. People find it very difficult to remember long and complex strings of random characters. This is why they choose the easy option. If you enforce long passwords, you instantly make yourself unpopular with staff. You also have the inevitable scenario of users sticking their passwords to their monitors to make it easier for them (and everyone else in the office) to get at them when they are needed.

So you have to trade-off security with the human touch. This is not a perfect solution, but it is the best you can achieve in a short amount of time. Start small, and people will get used to it over time. This will make your implementation of a strong password policy easier. Make the passwords expire after a month, and then increase the length of the required password by 1 character until you reach your desired password length. Password auditing is enabled on the mail server, so you have no worries about bad password choices.

Instant implementation of a strong password policy is the hardest thing you can ever do, it encroaches on users' work, and makes their lives more difficult and stressful. This is why I wrote users passwords down for them and advised them to change them afterwards.

> As bad as passwords are, users will go out of their way to make it worse. If you ask them to choose a password, they'll choose a lousy one. If you force them to choose a good one, they'll write it on a Post-It and stick it to their computer monitor. If you ask them to change it every month, they'll change it back to the password they changed it from last month. One study of actual passwords found that 16% of them were 3 characters or less, and 86% of them were easily crackable. Other studies have confirmed these statistics.

I also informed the person whose credit card details were logged that it would be a good idea for him to contact his credit card company, and inform them of the possibility that someone else could be using it without his knowledge.

I then pointed out to him that sending credit card details in an unencrypted e-

mail is basically the same as giving your credit card to a stranger in the street, except for the fact that there is no need for a signature if you purchase anything by credit card over the Internet or telephone.

After I finished checking the FreeBSD boxes and dealing with the password problem, I now had to get the gateway machine back up and running.

I installed RedHat 6.2 again on the gateway machine. Before connecting it to the Internet, I set up **Tripwire**; and after creating the initial MD5 database, saved it to a floppy for safekeeping. Then I started the machine up in single-user mode and gave it an IP address different from its original one, this would minimise the chance of someone from outside immediately trying to connect to it. After doing this, I ran **Rhnregister** and **Rhupd** to update the software installed on the system. When the machine had finished updating itself, I then installed and ran the **Bastille Linux** hardening utility to lock the machine down.

Once **Bastille** had finished locking down the machine, I used my **Chkrootkit** floppy disk to do a scan of the machine and make sure I had not been rooted again. (it can happen, believe me!)

> Between April and December 2000, seven default installations of RedHat 6.2 servers were attacked within three days of connecting to the Internet. Based on this, we estimate the life expectancy of a default installation of RedHat 6.2 server to be less then 72 hours. The last time we attempted to confirm this, the system was compromised in less than eight hours. The fastest time ever for a system to be compromised was 15 minutes. This means the system was scanned, probed, and exploited within 15 minutes of connecting to the Internet.

As an added layer of protection, I installed and enabled **Portsentry** to run on startup. This would block any incoming scans by adding a deny line for the scanning IP address to my firewall configuration once the first few ports on the gateway were scanned.

I decided to use the off-the-shelf **Default-to-deny** firewall provided by **Red Hat** to get the machine up and running in the fastest time. Any further ports that needed opening, I would evaluate as and when required by users. I would also tweak the rule set as soon as I had enabled the firewall to close any un-needed ports.

Finally, I ran **Tripwire** again, verified what changes had been made, updated the database and then copied that to a floppy disk. The disk was then write-protected, labelled, and placed in a safe location so I could rely on it for future checks.

Before continuing further, I took a dd snapshot of my Hard Disk's Linux partition

and burned this onto a CDROM. By doing this, I ensured having a good backup of the system in its original state to speed up any future reinstalls.

After confirming that my machine was secure and free from intruders, and having a good backup copy of its Hard Disk. I restarted the machine in multi-user mode. This enabled **Bind** in forward-only mode so internal users could have DNS access.

Cleaning up

All that remained to do now was the small task of scanning every machine on the internal network for any Trojan programs or viruses, which may have been placed there by the intruder.

Luckily, after joining the company, I had made sure that every windows machine had been unbound from **Netbios**, and was using **Netbeui** as a file-sharing protocol. This is a good thing to do, as **Netbeui** is not transportable across **TCP/IP**, so file-shares across the network are not visible to anyone connecting via **TCP/IP**.

Removing Netbios from TCP/IP in a Microsoft Windows environment…

To do this, you should open network properties by right clicking on **Network Neighborhood**, then select properties. Next select each and EVERY instance of **TCP/IP** in order, and perform the following:

Select “Bindings” and uncheck everything in the bindings box. Windows will complain that you have not bound **TCP/IP** to anything, but do not worry; you are going to use **Netbeui** instead. Click No when Windows asks if you want to select a binding. Then click OK.

Repeat this for the other **TCP/IP**-bound devices (even dial-up-networking. Windows will suggest that you do not need to change anything, but do it anyway, otherwise **Netbios** will still be bound to all **TCP/IP** instances).

When finished, you must now install **Netbeui**. Do this by selecting **Add protocol**, then select **Microsoft**, then finally choose **Netbeui**, then after clicking OK, reboot when requested.

After rebooting, check your **TCP/IP** bindings, and you will find that **Netbios** is now unbound from **TCP/IP**.

Congratulations, your shared files are no longer accessible via **TCP/IP**.

Unfortunately, Microsoft, in their wisdom have decided to omit **Netbeui** from Windows XP, however, should you wish to, you can still install the protocol by visiting the Microsoft Windows Support site.

To scan the Windows machines for Trojans and viruses, I chose Trend Micro's **PC-Cillin,** as I have never had any problems with it. In the past, it has picked up viruses and Trojans that other scanners with up-to-date signatures have missed.

I booted each machine from the **PC-Cillin** boot-disk set and scanned for malware. Luckily none were found. It seemed the intruder was happy enough to have just owned my gateway machine, or maybe I was lucky enough to find out in time that I was compromised!

After ensuring that the Antivirus pattern files on all the workstations were up to date, and double checking with http://windowsupdate.microsoft.com to confirm that all machines were current with security updates. I could inform the management that the system was up and running again and had been given the all clear.

I also requested that the company purchase a stand-alone tape drive so that we could perform regular backups of mission critical machines in future.

Conclusion

After 14 Hours of constant work from the first notification that the system was compromised, I had the system back in working order and secure once again.

When quantifying how much this breach of security had cost the company, I am not going to comment on dollars and cents, but bear in mind the costs involved from 14 hours of lost company time with no Internet access, no e-mails to & from customers, no orders taken for deliveries, and no correspondence with sub-contractors. In addition, costs arising from delays in answering e-mails because of the amount of downtime and new password policies being put in place, plus revenue lost from these delays. Last but not the least, the 14 hours (per person) of system admin activity (with overtime) plus assistants running around for the whole time, (which is not cheap if the job is done by a third party) and you have a scenario that can put a small company in the hands of the official receiver's office.

A simple intrusion into a company network can cripple, or in the worst cases, kill a company.

A secure machine should have all current security patches in place, be locked down, so no unnecessary ports or daemons are available for an intruder to connect to, and should have an Integrity Checker running, so any changes to files are noticed as soon as they are made. As well as this, Backups are vital in any networked environment to minimise the amount of downtime incurred by loss of data or system integrity.

A typical gateway machine should have nothing extra installed on it other than the packages required for it to perform its function as a gateway. This makes it more difficult for the intruder to use the gateway for his purposes once it has been compromised. If you have installed network monitors and sniffers to enable you to troubleshoot your network throughput, an intruder, immediately upon gaining control of the machine, can use these to do his own sniffing. As an added security measure, remove all compilers from the machine too. This means the intruder is unable to compile packages from downloaded source code.

If you have one available, connect a cheap printer to the gateway machine, so it prints out all logs immediately as well as saving them to disk. Also consider remote logging.

There have also been recent advances in Honeypot technology for use as an indicator of pending attacks. If you have the time and resources, consider adding a Honeypot to alert you of any suspicious activity. (It can also be configured to automatically add a deny line to your gateway firewall for the attempted hackers IP address.)

Your gateway is your last line of defence; you should be watching it carefully.

Nobody can ever be 100% sure that they will be immune to an intrusive attack. In fact you are more likely to be intruded upon at some point (be it by viruses or a real intrusion attack) than you are of being left untouched.

There are various documents available on the Internet contributed by extremely well informed people that back this argument up. If US Government and Department of Justice sites are vulnerable, then why are you so safe? These departments have millions of dollars at their disposal for security resources, but they still get hacked.

My point in all this is not to make you (too) paranoid; it is more to make you aware that you are not invulnerable just because you have all the latest patches on your system. Over time, new exploits are discovered and used by the hacker community. Upgrade as soon as these are announced. Keep an eye on your system and never become complacent. A little paranoia is a good thing amongst system administrators.

List of references

1: Betts, Bill. “Digital Forensics.” Information Security Magazine. March 2000.
URL: http://www.infosecuritymag.com/articles/march00/cover.shtml

2: Rude, Thomas. “DD and Computer Forensics, Examples of Using DD within UNIX to Create Physical Backups.” August 2000.
URL: http://www.crazytrain.com/dd.html

3: Miller, Toby. “An analysis of the t0rn rootkit.” No Date Given.
URL: http://www.sans.org/y2k/t0rn.htm

4: Hawkins, Sunny. “Understanding the Attackers Toolkit.” 13/01/2001.
URL: http://www.xenosystems.org/documentacao/geral/toolkit.html

5: Kendall, Kris & Kornblum, Jesse. “Scan of the Month – March.” 23/03/2001.
URL: http://project.honeynet.org/scans/scan13/som/som42.txt

6: URL: http://www.tripwire.org/downloads/index.php

7: URL: www.chkrootkit.org

8: URL: http://www.adel.nursat.kz/apg/

9: Schneier, Bruce. Secrets & Lies, Digital Security in a Networked World. Reading: Wiley & Sons. 2000.

10: URL: http://www.bastille-linux.org/

11: The HoneyNet Project. “Know your enemy: Statistics.” 22/07/2001.
URL: http://project.honeynet.org/papers/stats/

12: URL: http://www.psionic.com/products/portsentry.html

13: URL: http://support.microsoft.com/default.aspx?scid=KB;EN-US;q301041&

14: URL: http://www.trendmicro.com/pc-cillin/products/

15: The Honeynet Project. “Know Your Enemy: Honeynets.” 11/05/2002.
URL: http://project.honeynet.org/papers/honeynet/

16: Aardvark Daily. “Hackers Breach US Military Security.” 06/10/97.
URL: http://www.aardvark.co.nz/n294.htm

17: US Dept of Justice. “Wisconsin Hacker charged with military break-in.” 13/08/99.
URL: http://www.usdoj.gov/opa/pr/1999/August/387crm.htm

18: 2600 online magazine. “US (Japan’s) DOJ Homepage hack.” 17/08/96.
URL: http://www.2600.com/hackedphiles/doj/hackdoj_2600/

19: USDOJ Press release. “Juvenile hacker cuts off FAA tower.” 18/03/98.
URL: http://www.usdoj.gov/criminal/cybercrime/juvenilepld.htm