# UNIX Postmortem, What to do After You've Been Hacked.

Mark Henman

November 10, 2004

**Abstract**

For most system administrators, there is no question that at some point at least one of their systems is going to be hijacked by someone else. There are a number of resources to help people harden their systems against intruders, this document discusses steps to recover a system that has already been broken. While not a comprehensive reference on system inspection or recovery, this document should provide enough information to help an administrator quickly and successfully recover from an attack.

# Contents

# 1 Discovery

## 1.1 Realizing That You've Been Hacked

One of the most surprising statements that I hear from people who's computers have been broken into is, "It took quite a long time to discover that someone had compromised my system." Some crackers[1] leave obvious signs that they've attained access into the system. They'll alter web pages, change the login banner, send you crude messages while you're logged-in, or other nuisance activities. But many crackers are now gaining access to other people's systems, leaving a back door into the system, and then hiding. Sometimes hiding for months.

One of the main reasons for this is that there is a growing market for computer resources[1] for spammers, p2p file sharing systems, or denial of service attacks. Simply put, hijacked computer systems make wonderful worker nodes in larger networks dedicated to illegal activity. For example, if a spammer wants to sent a few thousand e-mails, but doesn't want to have his own IP address blacklisted by the mail servers that receive the spams, then he can use hijacked computers to send the e-mails for him.

Or, imagine a disgruntled employee leaving a mid sized company and acquiring the use of 10,000 hijacked home PCs for an evening to launch a massive distributed attack on his former employer. Believe it or not, these scenarios, and other like them, occur. And as more and more home computers are always on-line, the marketplace that supports this behavior expands every day.

Because of this growing tendency for crackers to remain hidden but retain the ability to take control of a system at will, it becomes increasingly important for a system administrator to be able to detect that a system has been compromised. The longer a cracker has access to a system, the more he can change it to better suit his needs. But he will also likely be probing for other vulnerable machines on the network, so the faster a break-in is identified and stopped, the less likely the intruder will have had time to get to other machines on the network. For these reasons, discovering a break-in right away is an administrator's best ally in being able to prevent a successful re-play of the attack.

## 1.2 Tools

As we will see below in the section "Trust Nothing", once a box is compromised you cannot guarantee that the reports you get from any tool are accurate. But by using several tools together

[1] Some people prefer to use the term "hacker" to describe someone who uses a computer for illegal activity. This document adheres to the FOLDOC definition and instead uses the term "cracker". See http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=hacker and http://foldoc.doc.ic.ac.uk/foldoc/foldoc.cgi?query=cracker

you are more likely to find out quickly when a break-in occurs. Because of the fluid nature of Internet attacks, it is best to have a variety of tools and to use them frequently to check for a growing amount of possible break-in evidence.

In this section, I present three tools, seccheck, chkrootkit and Tripwire, that may help identify a system that has been compromised. But even using these three tools together is not enough to ensure that a system is un-damaged. Rather that try to provide a comprehensive list of available tools, I leave it to the reader to find other tools that are appropriate for the systems that he or she is responsible for. Use Google or another search engine to research what tools are available for intrusion detection.

I recommend everyone fully research and evaluate Snort[2] and Tripwire[3] because they represent two distinct and powerful classes of tools: Network intrusion detection and File system intrusion detection. In addition to these, system administrators should have a good knowledge of resources available from the SANS Institute[4] or other reputable organizations that track security threats and attacker trends.

### 1.2.1 seccheck

Originally written by SuSE, this is a set of scripts that are executed daily, weekly, and monthly by cron to report system changes and other possible system insecurities. Included in seccheck's reports are:

- Changes in loaded kernel modules
- Checksum changes to executables in /bin, /sbin, /usr/bin, and /usr/sbin
- Files in user's home directories that are executable.
- All setuid and setgid executables.
- Changes in TCP/IP ports that are listening for incoming connections.

seccheck is not infallible, but it is a good start. If you use it, make sure that the reports it e-mails to the system administrator are read, and any ambiguities it finds are verifies for legitimacy.

Note that while seccheck was written by SuSE, most if not all of the scripts in it are portable to other Linux distributions as well as other UNIX variants.

### 1.2.2 chkrootkit

chkrootkit[5] (Check Root Kit) is a tools that looks for a number of signs of system compromise. While it won't identify a carefully executed attack, it will identify most "rootkits put in place by script kiddies".

Fortunately, more and more crackers today are not very skilled at what they do. Often they run scripted attacks across thousands of machines at a time looking for the latest known vulnerabilities. These attackers are called "Script Kiddies." They don't cover their tracks as well as a careful and well educated cracker would, but they don't need to. Instead they rely on massive attacks on large numbers of machines to find the occasional easy-to-crack system.

Consider this, even if only 1/2 of 1% of the machines directly accessible from the Internet are vunerable to a particular attack at any given time (a *very* conservative estimate), then a script kiddie can attack 10,000 machines in one weekend and expect to have 50 new additions to their list of hijackable systems. Usually, all the initial attack script will do is add the system to a list of targets and then install a root kit. A root kit is simply any tool that allows the attacker to get back into the system and become root. Once root access is gained, the attacker can do anything he wants.

This is where chkrootkit comes in. It performs over 100 different tests to looks for more than 50 different root kits. It won't remove them, but it will tell you if it finds something wrong and usually tells you where to look to get help.

### 1.2.3 Tripwire

Tripwire is an incredibly valuable tool that tracks and reports changes to important files. The program monitors key attributes of files that should not change, including binary signature, size, and expected change of size.

There are two versions of Tripwire available, an Open Source version and a commercial version. The commercial version has a larger feature list and offers more support besides just on-line documentation and news groups.

Tripwire works by monitoring not only files that are not expected to change, for example the executables in /bin, but it also watches files that normally grow in predictable ways, like system logs. It can also be made to only report on changes to specific attributes of some files. For instance, the contents of /etc/shadow change every time a user changes his password, so watching for changes to the contents of that file would be rather fruitless. However you you absolutely want to be notified if /etc/shadow's permissions or ownership changed.

Note that for any file system intrusion detection system to work properly, the database of file attributes that it uses to check against for changes must be kept secure. Some people put this database on a read-only file system like a CD-ROM and re-burn a new CD when legitimate changes are made. This requires more work on the administrator's part, but it is the only way to ensure that the reports from the tool are accurate.

### 1.3 Observation

Perhaps the best tool in the defense of crackers is simply spending time on the systems you are responsible for, and investigating things that seem out of the ordinary. Obviously, in an installation of hundreds of servers, it is impossible to spend much time observing every system. But for those administrators who routinely only work on a few systems it is imperative to know what "normal" behavior of the system is. The reason that this is so important is that many times a cracker will replace a few core commands that are frequently used with altered versions that re-open back doors into the system if they are discovered and closed. Along with re-opening doors, the commands /bin/ps and /bin/netstat are often replaced with altered versions that not only pave the way for a cracker to get into a system, but they also try to hide the presence of those back doors. An altered /bin/ps might not list all the running processes in a system. An altered /bin/netstat will hide listening TCP/IP ports.

Believe it or not, this is one place where the inconsistencies in differing versions of UNIX is actually an asset. If a cracker makes his modified version of netstat from BSD UNIX's source, then compiles that for Linux and installs it on a user's Linux system the output will look slightly different then the original binary would have produced. The columns might not line up the same or a command line switch might be different. An administrator who knows what the normal output from a command will be more likely to notice a change than one who occasionally works on a particular system.

One more important point about system observation. Sometimes the best place to observe a system is from the outside. Tools like nmap and saint can find listening ports from the outside of a box that would not be reported from an altered netstat. You can use some of the same tools a cracker would use to break into your systems to check and see if they've already gotten in.

## 2 Trust Nothing!

If you remember nothing else from this document, remember these two words: "Trust Nothing." Once a system has been compromised, you cannot trust anything the system tells you about itself.

Any command on the machine may have been replaced or the kernel itself may have been altered. A system that has been broken must be re-built from a trusted source. Do not try to recover only the commands that have been changed. You won't get them all. Even logging onto the system may send your username and password to some Internet site that will then use that information to try to attack other systems on your network. Once a machine has been compromised, it must be isolated, dissected, and re-constructed.

## 2.1 Disconnect From the Network

As long as the machine is connected to a network, there is the possibility that it is actively trying to reach out to other systems, whether your own or someone else's, to either report on knowledge gained from your system, or to try to break other systems close by. This should be your first action even if you just have a suspicion that the machine has been attacked and have not confirmed it.

This is important: physically unplug the network cable. Do not trust a software shutdown of the network interface, pull the plug instead.

Of course, in order to do anything with the system with the network, you must have direct console access to the machine. but you'll need that for the next step anyway.

## 2.2 Shutdown the Machine

Once you determine that the machine has in fact been breached, your next step will be to shut the machine off. This may be the only time that the following statement would ever apply:

*Issue a sync command to flush the file system buffers. Then, try to gracefully shut down the system, if anything doesn't look right, pull the plug.*

Yes, that bit of advice just went against the grain of everything you've ever heard about administering a UNIX system. The reason you need to pull the plug is simple: The machine may not actually shut down. I have seen two machines that have had their init command replaced by a root kit. On one system, the init command was replaced with one that always re-booted the system instead of shutting it down. On the other system, the last thing init did on a shutdown request was to schedule a system wake-up with the BIOS APM Manager so that the machine would power back on an hour later. The first case gave the administrator a false sense of security when he thought he had successfully shut down the system remotely. In the second case, the administrator saw the machine shut down and thought the rest of his network was safe. In both cases the machines were soon back under the control of a hijacker.

Shut the machine down, and if you're not going to immediately start re-building it then un-plug the power cord(s).

### 2.3 Boot From a Trusted Hard Drive

Once you've shut the system down, only bring it up by booting it from a trusted boot source. This can be a "rescue CD", or another hard drive that you know is safe. Booting from another hard drive that suffers from the same vulnerability will only let the attacker back in while you are trying to recover the system.

Note: Do not re-attach the network yet. Never bring the machine back up on the network until you have identified how the intruder entered the system and have removed that access point.

### 2.4 Mount the Compromised File Systems Without Execute Permissions

Once the system is booted from a good medium, mount the file systems from the old disk without execute permissions. This precaution is necessary because you don't want to accidentally run a corrupted command while trying to perform repairs.

## 3 Examination

This is where the real work (or art) begins. You now have to determine several things, how they affect the system, how they affect your business if the machine is a business machine, and what needs to change to make things safe again. Here are some things to look for while examining the system.

- How did the intruder get in?
- When did he get in?
- What did he change/delete?
- What did he see?
- Where else did he go on the network?
- How can a repeat attack be prevented?

## 3.1 Examine the Log Files

While a careful cracker will cover his tracks by removing incriminating log entries, lazy or careless crackers will leave a trail in the system logs of how they got in and what they did while they were there. Look in older log files to try to identify normal patterns in behavior. Also look for larger than normal log entries in short amounts of time.

For example, if you run a web server that normally serves one to two requests per minute but you see sections in the logs with hundreds or thousands of requests in a few minutes or seconds, that's a clear sign of attack. It may not be a sign of a successful attack, just an attempt. Be sure to examine the log entries just before and after these attacks to see if there is any indication that the attack was successful.

It is always recommended that system administrators review log files on a regular basis, irregardless of suspicious activity. This will familiarize them with normal system usage and logging patterns for each system that they administer. As they become more familiar with the system, they will be better equipped to identify abnormal behavior.

## 3.2 Look for Changed Files

Most intruders will want back in once they've gotten in. In order to do this they will change as much as they can on the system that will make their job easier the next time. They will not just alter a single command. They will alter as many as they think they can without sending up too many warnings.

### 3.2.1 Compare to a Trusted Source

If you were diligent in your duties, you should have a reliable full system backup from before the break-in. You can use this as a source to use for looking for changed system binaries. If you don't have a full system backup, then use the operating system install media, or another hard drive that has not been compromised for the comparison. Be sure to check the boot sector, kernel, kernel modules, executables and shared libraries.

Make a special note if /bin/login or /usr/bin/passwd have changed. If they have then you must assume that your password and your user's passwords have been harvested. While it is always recommended that all users change their passwords regularly, it is essential that they do it after a break-in. It is even more vital for the security of the entire network that all passwords are changed if you think that a password harvester was used on one of the systems. You don't have to be in any

computing field very long to realize that people re-use their passwords. No matter how often they are told that they shouldn't do it, unless you put in measures to enforce distinct passwords, users will always re-use their passwords. Crackers know this, and they exploit it whenever they can.

Make a list of all the files that changed, then take that list to Google to see if anyone else has reported a similar set of changes to their system. This is a good way to lean on the experience of others to see how they hardened their systems from a similar attack.

## 3.3 Look for Viewed Files

One of the first questions that comes up after a break-in is to know what the attacker saw. This is sometimes more important than finding out what was changed. For personal PCs, common items for an attacker to look for are credit card numbers, bank account numbers, e-mail addresses, etc. For business systems, this may include customer information such as social security numbers, account billing information, corporate earnings reports (especially handy to have the day before the official public release of this information so the attacker can adjust his stock portfolio accordingly), etc.

The attacker may have simply opened the file on the compromised system, or he may have transmitted the file to another server on the Internet. You may be able to determine this by examining any router or firewall logs.

Going back to our first premise of "Trust Nothing", you should try to determine if the attacker had the ability to reset the last accessed time stamp for files on the file system. A common trick is to tell the system to sync its clock to a time server on the Internet that is intentionally incorrect. A clock re-sync is usually not considered a suspect activity in log scans, but it allows the intruder to alter files and make the time stamps appear to be outside of the time the intruder was active on the system. If the file system does allow modification of the last accessed time stamp, then you need to assume that every file was viewed. If the system doesn't allow it, then you will have a good chance of building a list of suspect files.

You may never be able to give a definitive answer to the question of what an attacker saw, but you can give an estimate. Begin by getting a list of the last accessed time for every sensitive file in the system with a last accessed time stamp later than the first possible attack time and date. (The definition of 'sensitive' varies based on the context of the system.)

It is better to err on the side of caution and assume that a file was viewed even if you are not 100% sure that is was. Only omit files from the list that you are completely certain that they were not viewed.

### 3.4 Look For Possible Back doors Into Other Machines on Your Network

Hijackers know that they will only have control of hijacked systems for a limited time. They will eventually be discovered and the doors to compromised systems will be closed. In order to be successful, they have to continually find more machines that can be controlled.

For this reason, you also need to check all computers that are accessible from the target machine. The hijacker will probably examine the /etc/hosts file and any domain name server files to discover the network topology and naming conventions for the network.

Any other machine that can be easily accessed because of a trust relationship between it and the compromised machine should be checked for intrusion. These include systems accessible via rlogin, ssh, samba, NFS, telnet, cvs pserver, SMTP or other network protocols; and yes, the list may include every machine on the network. Pay special attention to systems that share file systems or user accounts.

## 4 System Restoration

### 4.1 Backup the User Data, Carefully Checking for Possible Alterations.

First gather the list of files that have changed since the last known safe backup. Then, either you or your users should certify that any changes in those files were legitimate. Any executable binary files in the user's home directories are suspect for corruption, and should be treated appropriately.

This can be very tricky, not from a technical standpoint, but from a political one. The real trick is how you interact with your users and the possible removal of their files. Users generally don't want other people poking around in their files, even in the face of a security breach. If a user just finished spending 500 hours updating a financial report and then gets a call from an system administrator how states very plainly that his file contains a macro that opens a back door to the system, that user is not going to sit happily and watch you undo all the work he's done by restoring a three week old version of the file. There is no one right answer on how to handle this situation other than to be "carefully diplomatic."

### 4.2 Re-install the Operating System.

- From a full system backup prior to the earliest possible break-in date.
- From original install media (safer).

If you use a Solaris Jumpstart server or a RedHat Kickstart Server, then reformat the hard drive, plug-in the network cable and rebuild the system from scratch. (Though you may want to verify that the Jumpstart server or Kickstart server isn't going to build a vulnerable system.)

If you don't use Jumpstart or Kickstart or another equivalent tool, then restore the system from a known good backup. If you don't have a known good backup, then re-install from scratch using the original installation media.

In every case, don't forget to re-build the boot sector.

If the machine has been highly configured from its last full backup, there may be a desire to overlay a known good operating system on top of the corrupted one. Someone may also want to try to merge the old configuration files into the restored ones. Don't fall into this trap. Yes, it will save you some time in re-configuring the machine, but you can inadvertently re-open a security hole that will invite the crackers right back in. Take the time to re-configure the system if you have to, and use it as an opportunity for an audit.

### 4.3 Restore User Data.

If everything went well while backing up the user data, and you feel confident that the user files are free of malicious code, then the restoration of user data should be a trivial step. Even so, this is a good time to double check that the users are following appropriate policy for file and directory permissions. Also, this is a good time to check for files such as .rhosts and others that may compromise system security.

## 5 Followup

### 5.1 Harden the System Against the Same Attack.

This topic is the subject of many other papers and articles. I will not discuss it further here other than to say that it needs to be done. Special attention needs to be paid to any vulnerable areas discovered from the steps performed above. If you simply restored the system from an earlier backup, then you *must* harden the system prior to re-connecting the network.

## 5.2 Check for Abnormal Behavior.

Often a good check to perform is to connect the system under question to a managed router that is not connected to the Internet or any other systems. (You might need one other system to monitor the router, but nothing that you wouldn't be able to live without for a while.) Now you can watch the router to see if the system tries to communicate with the Internet. You can also port-scan the machine to see if it is listening on any ports other than the ones you know it should.

Basically, go back and get those cracker tools and see if you can break into your own box. If you can, then go back to the beginning and start over.

## 5.3 Bring the System Back Into Service.

This is the "Moment of Truth." After you have verified that there are no unwanted programs lurking around, and the system seems to be functioning normally, then plug the network cable back in.

## 5.4 Watch Logs for Repeat Break-in Attempts.

If someone got in once, they will try it again. They will usually try the same method, then try other similar methods.

Please note that watching the system logs is not a way to stroke one's own ego to see how the machine is now resisting attacks. When the machine comes under attack again, look to see where the attack is coming from. This knowledge can be used to build a IP blacklist, set up connection filters or modify routing tables to help keep the machines safe.

Just remember that the machine that is currently performing the attack may have been hijacked itself, and origin of the attack may be somewhere else completely.

# 6 Conclusion

For the most part, the steps described in this document are simply common sense. The important thing to remember is to *quickly isolate* the system, then *slowly examine* and rebuild it. Discovering that a machine has been compromised can be infuriating, and many administrators that have not experienced a break-in before will react too slowly in isolating the system and then re-build it before fully understanding how the break-in occurred and how to prevent it. Preparing for the

break-in by having frequent back-ups and a plan in place can make the system recovery a much easier process.

# References


[1] ZDNet UK, *Asian spammers 'hijack broadband PCs'*, http://news.zdnet.co.uk/communications/networks/0%2C39020345%2C39117251%2C00.htm

[2] Snort, *The Open Source Network Intrusion Detection System*, http://www.snort.org

[3] Tripwire, *A tool that checks to see what has changed on your system*, http://www.tripwire.org

[4] SANS Institute, *SysAdmin, Audit, Network, Security*, http://www.sans.org

[5] chkrootkit, *locally checks for signs of a rootkit*, http://www.chkrootkit.org