CHAPTER 4

Pen-Testing Process

In this chapter, we will cover the non-technical and process aspects of ethical hacking.

- Difference between types of assessments
  - Penetration test
  - Red teaming
  - System test
- How to get started
  - Building a team
  - Building a lab
  - Contracts, safety, and the "get out of jail free" letter
- Steps to a successful assessment
  - Assessment planning
  - Initial customer meeting
  - Conduct of a test
  - Reporting out

Before we get into the technical aspects of ethical hacking, we need to talk about the overall penetration testing process and, really, how to be a professional hacker. There is more to a successful pen-test than swooping in, getting root, and handing the customer a report on your way out the door. Conducting an efficient vulnerability assessment is more involved than locking yourself and your buddies in a room with a stack of Mountain Dew, a couple pizzas, and a network drop. There is also a big difference between application vulnerability assessments and red teaming. This chapter will talk about those differences and how to do each well.

Types of Tests

The prototypical hacker—if there is such a thing—is skilled at finding the one vulnerability or vulnerable machine that gives away the rest of the network. This type of scanning, probing, exploiting, and escalating is called penetration testing. The primary goal of a pen-test is to "own" the network; the secondary goal is to own the network in as many different ways as you can so that you can point out each flaw to the customer. Pen-testing a network is an excellent way to test the effectiveness of an organization's security measures and to expose their security gaps.

The terms penetration testing and vulnerability assessment are often used synonymously in the security community. However, there is a difference between the two. A vulnerability assessment scans for and points out vulnerabilities but does not exploit them. Vulnerability assessments can be completely automated using tools such as ISS, Nessus, or Retina. Vulnerability assessments are great because, first, they're easy to do and, second, they show the host-by-host detail that, hopefully, shows every potential network vulnerability. A pen-test, on the other hand, focuses on the actual exploitation of the found vulnerabilities. In Chapter 6, we will show you how to use an automated tool (Core's IMPACT) that performs all phases of a pen-test.

Some customers, however, benefit more from a global look at their information security. This broad analysis of a customer's security is called red teaming. Red teaming not only includes the network surveying and port scanning, two components of pen-testing, but also testing of Internet-facing applications, IDS testing, social engineering, and recognizing security issues within an organization. Both red teaming and pen-testing (a subset of red teaming, see Figure 4-1) should include a complete report to be used as a checklist to fix vulnerabilities and should include separate outbriefs with both the technical staff who will be fixing the problems and also the executive staff who hired you to do the work. We'll provide outbrief guidance later in this chapter.

System testing is a third type of security service and is requested by customers interested in the security posture of a specific system or application, often a network-accessible system or application. This type of assessment is more of an art than pure pen-testing or red teaming.

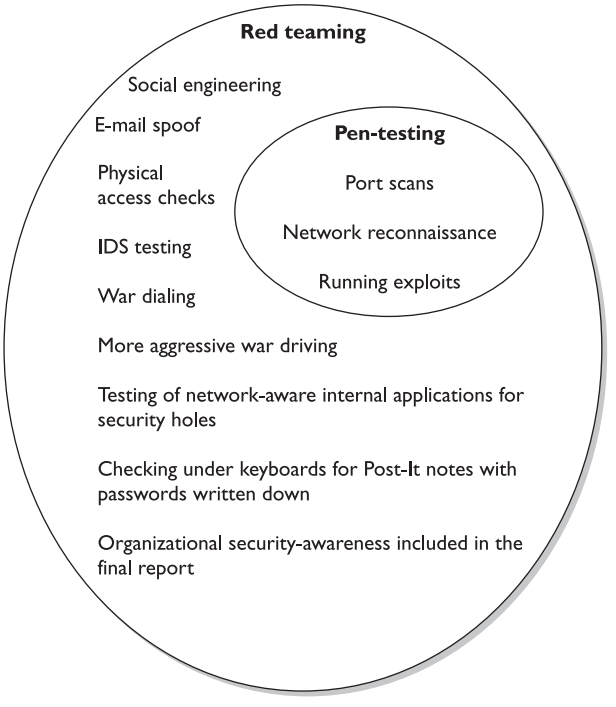

Figure 4-1 Differences between pen-testing and red teaming

Your job while system testing is not only to point out known holes; you also need to find new vulnerabilities by digging deeply into how the system works and pointing out incorrect security assumptions. A successful system test will often find, for example, buffer overruns when passing the application too much data or will point out the key file that is world-writable and used for making trust decisions.

References

Christopher Peake, "Red Teaming: The Art of Ethical Hacking"
www.giac.org/practical/GSEC/Christopher_Peake_GSEC.pdf

Ramping Up

We will talk about how to run each type of assessment, but let's first look at some things that you need to do before going out on a first assessment.

Building a Team

You'll often need more than only yourself to complete a customer-requested assessment in a reasonable timeframe. There are no hard-and-fast rules in team building, and the difference between the productivity of a good team and a bad team is vast so it pays to spend some time thinking about how to build a good team.

First and foremost, when choosing a team, put together people who are going to get along. This group will have to work together closely and synergy between team members greatly increases the effectiveness of the team. The last thing you want is team members getting on each others' nerves. This may seem like common sense, but nothing ruins a pentest faster than a bickering team.

Keeping a team happy and hacking is also easier when each team member knows his role. There are a few distinct team roles that you may or may not need to fill, depending on the type of assessment.

The Technical Lead (Tech Lead)

For each assessment, a senior team member should be identified as the tech lead. It is the tech lead's responsibility to run the assessment, directing all aspects of the job and directing the workflow. While the tech lead should certainly take input from everyone on the team, it's easy to resolve conflicting ideas on how an assessment should be run—the tech lead's way is how it will be run. Having someone identified in advance as the tech lead makes it easier to make snap decisions in the field.

The tech lead should be a veteran hacker, comfortable in a lead position, and capable of doing the entire assessment solo if given sufficient time. The tech lead's focus is team productivity and overall assessment quality. On some jobs with some teams, the tech lead can be heads-down hacking during the entire mission. Other times, the tech lead must forgo individual productivity to help direct other team members, ensuring the work queue is full for each person.

The Team Chief

Each mission that involves an external customer or travel should have an individual identified as team chief. The team chief is a facilitator, enabling everyone else to focus on the job. The team chief should be the primary interface with the customer, a huge (but important) time sink on many assessments. The team chief will also ensure that the team's administrative or logistics support makes travel arrangements for all team members. Anything administrative that can be done for the rest of the team should be done either by the team chief or arranged by the team chief to be done by the administrative support. The team chief should work with the customer and the tech lead to establish hours of operation and other scheduling issues. Finally, the team chief should participate as a team member when not otherwise occupied.

On small or experienced teams, a single person can be both the tech lead and team chief. However, it is almost always better to have a separate tech lead and team chief. The team chief is often called away to give the customer status updates or to find the answer to a question a team member asks. The team chief does not have to be the most experienced team member. In fact, a junior hacker can be an effective team member if he or she can handle the organization and customer interaction required of the team chief.

Team Members

The tech lead and team chief should work together to assemble an effective team of individual contributors. As you might guess, a team with diverse talents is often the more productive team. Especially when pen-testing, having a team member focus solely on a single area while knowing little about other areas is fine. The important consideration is coverage, making sure you have at least one person who is comfortable with any system the team will be testing.

Building a Lab

No matter what type of assessments you'll be performing, you will need a lab of targets to use for experimentation and research. If you plan to primarily attack Windows, you'll logically want a lab full of Windows boxes of different OS, service pack, and patch level. Because each passive target will be doing next to nothing while it waits for your attack, you can easily get away with making each target a virtual machine. If you buy a couple of beefy servers, you can run eight (or more) virtual machines per physical server, saving on space, heat, and cost. Another benefit of using virtual machines rather than physical machines is ease of rollback. You can trash a VM and simply roll back all changes and reboot. Running terminal services on all targets will even eliminate the need to be physically at the console.

You'll also need some kind of system to flag certain machines as being in use and to track the active operating systems on each virtual machine and the additional images available for each machine. One system that has worked well for other teams is to periodically back up all systems to a central machine and use that central repository as a version control system. Something as simple as a text file on the central machine's desktop used to check out machines to different people should work fine for small teams.

The text file could also list the current OS version on each VM and the images available for use. Larger teams may want to devise a more sophisticated solution, but you can start with something simple and ad hoc.

Contracts, Safety, and Staying Out of Jail

You must not, however, start out with a simple, ad hoc contract for your engagements. Working on a customer's production network is a risk with every packet that leaves your hacking machine.

Before any work is done for any customer, a few things must be in place. First, the customer must completely understand the types of tests your team will be conducting. This can be more difficult than it sounds when you work with customers that are not tech-savvy. Your team chief must explain the good, bad, and the ugly of what could take place during the tests. You will, of course, be taking every precaution, but using some exploits may freeze or reboot vulnerable systems. The customer needs to understand the potential negative impact this could have on their network. Given the risks, they may choose to have tests done on the weekend or during off hours.

The next thing that must be in place is a solid contract and a document that outlines the scope of the assessment, including the specific tasks that will take place. These documents protect your assessment team and should be drafted and/or approved by a lawyer, preferably one who specializes in cyberlaw. Be sure to submit the contract to the customer with sufficient lead team for their corporate legal representative to review and approve the contract. The importance of both these documents cannot be overstated. There are documented cases where security professionals cracked customer's passwords (a common testing procedure) without the customer's knowledge and without password cracking being listed as part of the contract. When the customer found out their employee passwords were being uncovered, some of the assessments were stopped and the team was fired; in one case, a security professional was even arrested. It is essential to take the time to explain to your customer all things that will take place during your test and their possible consequence.

NOTE It is also important to get the proper level of management to sign the contracts. Make sure that the person you are dealing with has the authority to allow and authorize this type of activity to take place with the customer.

Depending on the type of tests performed, you may need to be aware of additional concerns. The company's employees may react in a defensive nature (as they should) if you are truly emulating an enemy by carrying out stealth-like activity such as social engineering, testing of physical security, and testing the reaction of the security team. Your team members may have the police called on them or, worse, if they are sneaking around a building looking for easy entry points into the facility, the security guard and/or dogs might not react in a friendly and welcoming nature.

To ensure your team members are not taken away in a police car, each member should always have a copy of an authorization letter with them. This letter is a signed document indicating that the customer's management staff is aware of and approves these types of activities. If you do not have these documents on your person when a security guard catches you doing something normally illegal, your story that their boss is paying you to do it will not carry much credibility. In the industry these signed documents are commonly referred to as "get out of jail free cards."

Assessment Process

The overall process of any type of organized ethical hacking is fairly straightforward. You footprint your target, probe for vulnerabilities, exploit those vulnerabilities (if the negotiated rules of engagement call for exploitation), and then share the results with your customer. As we talked about earlier in the chapter, however, each type of assessment has a different purpose with different parts of the assessment being more important. In this section, we'll first talk about what every assessment should include and then we'll outline the process for pen-testing, red teaming, and system testing.

Assessment Planning

After you've built your team and have a solid contract and authorization letter ready, it's time to start thinking about the individual mission and customer with whom you are working. You should plan and negotiate the length of engagement and number of team members with your customer to get sufficient coverage of all the targets the customer is interested in you assessing. Telling the customer on the last day of the engagement that you only assessed half the targets they are interested in indicates poor planning on your part. Solid estimates up front and frequent customer updates during the assessment will help minimize the chance of this happening.

During your assessment planning, you should also make sure that your customer understands the type of test they'll be getting and the type of results they should expect from you. If they're primarily interested in securing their web applications, a focused system test of the web applications from the Internet is far more appropriate than a general pen-test of their corporate network from inside their firewall. It's important to emulate the type of attacker they're most interested in from the network location from which they are most concerned. You would likely spend some time inside the firewall as well but not spend a great deal of time trying to penetrate their office network.

NOTE It is often part of the job of the security professional to help the customer understand the type of threats the company faces. Customers usually have a naïve and narrow view of their network's enemies.

Finally, when working from your customer's location, you should schedule a kickoff meeting the morning of your first day of work to meet your customer, answer any questions they have, and set expectations for the assessment.

On-Site Meeting with the Customer to Kick Off Assessment

During your assessment kickoff meeting with the customer, the team chief should stand up and give a spiel about your team and what you will be doing. You don't need to go into every single tool that you'll be using, but it's good to give the customer an overview of the process you'll be following to give them confidence that you know what you're doing. Gauge their response and interest to decide whether to go deeper into each topic or to move on to the next. It's handy to have a PowerPoint presentation for the initial meeting but it's not essential. If you're comfortable doing so, invite the customer's network folks to come by any time to shoulder-surf. Don't be secretive about what you're doing. Try to build your customer's confidence during this kickoff meeting. Offer to give periodic updates and set up a schedule for giving those updates if the customer is interested. Try very hard to make the entire process customer-centric, focusing on their priorities. Finally, tell the customer what type of results they'll be getting and when they should expect to see the results. If it's going to take a couple months to build up your final report, tell them that up front.

The kickoff meeting is largely non-technical so a team chief who has been through a few assessments can drive the meeting alone, leaving the tech lead and team members to get to work right away. From here, each of the assessment types have a different process. We'll discuss each, starting with the pen-test.

Penetration Test Process

We mentioned earlier that the goal of the pen-test is to gain privileged access on as many targets as possible. While that is true, the actual reason for the pen-test is to help your customer secure their network or system. That's the reason you should spend the time and energy negotiating the important targets with the customer and spend time talking with them before, during, and after the assessment. Try to keep the customer's priorities and objectives in mind when performing a pen-test, especially, as it's easy to get sidetracked.

Discovery

The first step in the pen-test is target discovery. Given no inside information, you want to discover as much about your targets as you can. This is usually called "footprinting" and is an important part of the attack because it simulates the way an unauthorized hacker would start an attack. Before jumping straight to the ping sweeps and port scans, it's interesting to see what you can find without sending a single network packet to the target. This activity is referred to as open source research. The whois and ARIN/RIPE/APNIC databases provide a wealth of information including IP ranges, name servers, and potential usernames listed as contacts. You might also want to query Google for interesting information about your target. If you include the "site:" keyword in your search, Google returns only results from the target domain.

This type of open source research can also be automated with a tool developed by James Greig called **dmitry** (Deepmagic Information Gathering Tool). It is a command

line utility that runs on Unix, Linux, and BSD variants (including Mac OS X) and pulls whois information, Netcraft data (www.netcraft.com), searches for subdomains, and port-scans the target. To run **dmitry** without the portscan, use the following command line:

```
GrayHat-1> ./dmitry -iwns mit.edu
```

dmitry creates a file named (in this case) mit.edu.txt that gives a wealth of information about MIT's network.

After you've gathered as much information as possible anonymously, it's time to get a bit more intrusive and find which hosts in your range of possible targets are alive. A simple ping sweep might be enough or you may need to do something more clever to enumerate through a firewall. Regardless, you need a solid list of live targets. Once you have that list, find out on which ports your targets are listening. There are many port scanners that will give you this information and we present another (**scanrand**) in the next chapter. With a list of live targets and open ports, the tech lead can begin directing the workflow, passing off a list of IPs running SNMP to the infrastructure expert, those listening on TCP 139 to the Windows team member, and so on. From here, we transition from the discovery phase to vulnerability enumeration.

Vulnerability Enumeration

Each open port indicates a running service and many services have known vulnerabilities. This vulnerability enumeration phase involves matching up open ports to running services and then to known vulnerabilities. This phase is more intrusive than the discovery phase and should result in a tidy list of systems that can be exploited for some level of access.

This enumeration involves actively trying to pull service banners, sniffing credentials on the wire, enumerating network shares from NetBIOS information, and pinpointing unpatched operating system components. Most hacking literature focuses on this and the next phase so we're going to gloss over it a bit. There are many exploits for many different services. The important point to make here is the methodical process that an ethical hacker should follow on a pen-test.

NOTE As stated in Chapter 1, this book is really a "next generation" ethical hacking book. There are several books that adequately explain how to exploit services so we have chosen not to cover these issues in depth.

Exploiting Mapped Vulnerabilities

After you have a list of systems you think are vulnerable to various exploits, it's time to prove it. On a pen-test especially, it's important to actually penetrate, gaining user and eventually privileged access on as many systems as possible. The goal of the pen-test is to point out your customer's security gaps. Those gaps are illustrated more forcefully if you can show that you "own" every box on the customer's network or that you have unfettered access to information considered to be a "golden nugget." There's something magical about showing the CEO a screenshot of his e-mail in-box or an Excel spreadsheet of employee salaries. Security quickly becomes top priority when you can turn the

potential threat of a potential network attack into real-life consequences of a real-life penetration.

Advantages and Disadvantages of Pen-testing

Pen-testing is a fantastic method for raising the security awareness in a company and showing how easily an attack can happen. It is a good way to point out the handful of flaws that allow an attacker to escalate from an anonymous outsider to an all-powerful administrator or root user. If kept somewhat secret, it can be a good way to measure the IT staff's awareness and response to an attack in progress. And finally, pen-testing is a great way to secure funding for more security technology, training, or third-party help when the management team sees the effects of an attack firsthand.

However, pen-testing does have its limitations. Even a very successful penetration test might leave scores of vulnerabilities unidentified. The best way to secure a network is really not through pen-testing. Rather, the best approach is to do a vulnerability assessment that uses the broad stroke approach to list every potential vulnerability and then follow up those results with a penetration test that attempts to exploit the identified vulnerabilities. An organization should not develop a false sense of security after fixing the holes the pen-test identified; there may be many more.

References

Dave Burrows, "Introduction to Becoming a Penetration Tester"
www.sans.org/rr/penetration/101.php

Dmitry Homepage www.deepmagic.org/tools.htm

Scott Granneman, "Googling Up Passwords" www.securityfocus.com/printable/columnists/224/

Red Teaming Process

While pen-testing is great at showing how deeply an attacker can get into a network, red teaming should show all the ways an attacker can get in. The term "red team" is borrowed from the military. Military training exercises call the good guys "blue forces"—or simply "blue"—and they call the adversary "red forces" or the red team. (The exercise is observed, refereed, and evaluated by the "white team.") So red teaming is simulating the adversary. In this book, we use the term red teaming not just as simulating an adversary but simulating a covert adversary, skilled in the art of exploiting systems and social engineering.

There are different philosophies on red teaming but to properly simulate the adversary, a red team should not be given a network drop and an office across the hall from the IT team. A good red team should find its own access onto the network and, if possible, remain hidden throughout the engagement. This is a great exercise for the customer because many different aspects of network security are tested, including incident response.

Red teaming can be methodical or it can be more ad hoc. During the pre-planning phase of the assessment (discussed earlier), you should tell the customer about your red

teaming capabilities and have them explicitly outline the areas they would like tested. After you get everything lined up, the phases of a red team assessment are typically about access and privilege.

Gaining Network Access

You first need an access vector onto the network. There are many ways in, so this is easier than it sounds. You can try penetrating through the external firewall, compromising a machine on the inside, and using that machine to launch the rest of your attack. This is like a pure pen-test and proceeds as outlined earlier.

If you can't get through the firewall, you need a different vector onto the network. The most common are wireless access points, modems connected to the corporate network, public terminals, and social engineering. Wireless access points really are the most convenient access for a red team. Simply drive around the customer's worksite, looking for wireless access points. Depending on the configuration of the access point, you may just be able to park and do the rest of the assessment from your vehicle or leave a "plant box" that you can SSH into from the hotel.

After you find your wireless access point and have secured initial access—or while waiting to crack the wireless security mechanism that is in place—look for additional ways in. Modems are used less often these days, but they are still around, pcAnywhere being one of the prime culprits. The hacker group THC makes THC Scan, a free "war dialing" tool. War dialing is the practice of dialing all the phone numbers in a range in order to find those that will answer with a modem.

Red teams should also test user awareness and the customer's physical security. Simply walk around the customer's worksite looking for unsecured workstations. Remember to keep your "get out of jail free" card with you whenever you are doing this! Guest workstations that you can sit at for hours unnoticed are especially attractive. Bring along a bootable Linux CD with all your attack tools or a few tools on a USB drive and you'll be in business. If you're a little braver, walk into an unused office, jack in a laptop with a wireless card and walk back out. Even with locked exterior doors, you will be amazed at how many people will hold the door open for you if your hands are full.

While it may not be the most enjoyable task, dumpster diving is a great way to find information that could expose weakness in an organization's security. Every day people throw away things they shouldn't. In the trash you can commonly find printed-out e-mails, documents describing security threats or vulnerabilities, passwords on Post-It notes, configuration information for installed systems, and CDs containing source code. If you need some token (vehicle pass or badge) to gain physical access to an organization, you can commonly find discarded passes or expired temporary badges in the trash. Creative use of a Sharpie can turn an expired pass into an active pass. At the very least, these tokens provide a pattern or sample you can use to build your own legitimate-looking access token.

There are many ways to gain access without much effort. On red team engagements, you want to test several different ways so you can report back to your customer the overall strength of their perimeter. As we talked about in the first chapters and again earlier in this chapter, this type of intrusive hacking is illegal without the protection of your authorization letter. Do not practice any of this on a live network without the protection of a contract signed by the authorizing management!

Escalating Privileges

After you have initial network access, the next step—just as in pen-testing—is privilege escalation. You can escalate using the normal pen-test process, but you should also test different aspects of your customer's security in parallel. To get into the true spirit of red teaming, don't grind out domain ownership the methodical way; instead, first try sending spoofed e-mail to the administrators asking for their password directly. To pull off something like this, you'll likely need a scam of some sort. One such scam is to build an official-looking web page, complete with company logo at the top of the page, asking users to enter their username and login password in order to test the strength of their password. Add copy to the web page stating the importance of a strong password and the company's commitment to information security through strong passwords. The entered passwords, of course, should be logged to a text file for you to use later. You could even build a small script or program that adds an account to the Administrators group and attach that program to a targeted e-mail with instructions to run the program. (Remember the ILOVEYOU virus?) Name it something appropriate for your target audience. Social engineering scams like this will get you more access than you might guess.

Incident Response Test

After you eventually have as much access as you'd like, it's time to get caught. Yes, you read right—you want to get caught. When no one expects a test team to be onsite, their actual incident response process can be tested. You'll want to test both their physical security incident response and their cybersecurity incident response, to include the reaction of the local CERT to the red team's actions.

To test their cybersecurity incident response, try to get caught by their IDS with noisy network scanning. If that doesn't work, try creating a new user account or adding an existing user to the Domain Administrators group to see if anyone notices. See if they notice you logging in interactively with a service account to a workstation (something that shouldn't normally happen). If they have port security enabled, plug a machine into several disabled ports to see if they're watching for unauthorized computers on their network.

To test physical security incident response and user awareness, be more aggressive in your social engineering. Ask for directions to the wiring closet or the server room. Try to get a visitor's badge without giving a purpose for being there. Call the helpdesk and ask to have an account created. Walk in the front door and ask someone if you can borrow the computer they are using for a bit. Set up your own wireless access point and antenna as conspicuously as possible and plug it into a network jack. See how many people you can get to log into and then out of a laptop you carry around.

You want to keep pushing to see how far you can get before getting caught. When you do get caught, show your "get out of jail free" letter and consider the incident response test complete. Explain to your customer in your outbrief how long it took to get caught.

Advantages and Disadvantages of Red Teaming

We presented only a few of the tools of red teaming. The more important takeaway of this section is the process and attitude of red teaming. Testing an organization that does not expect you to be there is a huge advantage because you can capture their everyday

security posture, not the heightened state when they are being pen-tested. Red teaming also tests much more than the actual network attack vector, giving the customer a real look at how serious hackers can hack their systems.

Red teaming, as we define it in this section, gives the customer a different look at their total network security. Again, it might not be the best way to secure the network. The vulnerability assessment shows the easy-to-scan-for vulnerabilities while a talented red team can point out holes in a network's security that can be just as or even more critical to the company.

System Test Process

Testing a system for new vulnerabilities is quite different from pen-testing and red teaming but shares some of the same process. The system test is more an art form than a process, and you can't follow a step-by-step checklist from start to finish. Some hackers find it more difficult because it requires more creativity; others find it easier because you can focus more tightly on a single product and its limited attack vectors. The big-picture steps of a system test are attack surface enumeration (footprinting) and focused exploitation.

In this section, we'll assume you are tasked with finding new vulnerabilities in a piece of software or a limited portion of an operating system. All references to "the application" in this section refer to the application or OS component you are testing for vulnerabilities. If you are testing a hardware system (a Bluetooth phone, for example) or even a more complex system (such as a commercial aircraft), the process is similar but you'll use different tools.

Attack Surface Enumeration (Footprinting)

During this phase, you will identify the application's attack surface area. The attack surface includes every single way the system interacts with external data of any kind. This investigation must be done thoroughly as the rest of the assessment depends on what you find here. There is more to identifying, for example, an operating system's complete attack surface area than pointing Nmap at its IP address. The network stack is a prime attack vector, but you should also consider any data stored or retrieved all the way from installation through use to uninstallation.

Footprinting the Application Installation The interesting part of installation is the "persistent state" the application initially sets up for later use. The complete state of a system is all-encompassing to include the entire content of memory and disk, but testing all that is very time consuming and actually not very useful. The most useful parts of system state to test are registry keys (for Windows systems), initialization files, machine policy, and file and directory permissions (for Unix) and access control lists (for Windows). The easiest way to check the state changes that happen during installation is to take a snapshot of these areas you are interested in before and after installation and look at the differences. You may want to do this kind of testing on a virtual machine so you can subsequently roll back to the previous state when you want to install a second time.

Snapshot utilities are cheap. One such utility is InCtrl5 from *PC Magazine*, which can be downloaded for $5. InCtrl5 quickly snapshots registry and file changes on Windows

before and after installation and reports the differences. You can do the same kind of checking with Tripwire, available for nearly all flavors of Unix and Linux for free.

You see crazy things during installation. Things like the installation program changing the permissions on the Program Files directory to be world-writable (gulp!) or placing configuration information for a program running as a privileged service in the HKCU registry hive, writable by a limited logged-on user. When examining the snapshot differences, pay special attention to the location and access permissions of created files and registry keys. If you can find an initialization file stored in a weakly protected directory, perhaps a local unprivileged user can change the contents of that to have a privileged process run arbitrary code for him.

Footprinting Normal Use After you have a reliable list of state changes introduced during installation, you'll want to see how the running process works. To footprint a passively running Windows process, you'll need three programs from Sysinternals: Filemon, Regmon, and Process Explorer. Filemon reveals all file system activity in real time. Regmon displays (again, in real time) all registry access made by your application. You'll likely see some of the same keys accessed as were created during your installation footprint, but there may be others as well. Finally, Process Explorer shows which DLLs, handles, and network ports your program opens. Process Explorer displays a lot of information so if you are only interested in network ports, you might instead use the command line **netstat** utility. As you become more experienced in the security field, using the Sysinternals tools will become second nature and you will reach for them anytime you want to know what is happening under the hood.

To prepare for the application's first launch after installation, launch Filemon and filter the results to display file activity originating from your application only. This is easy to do—just click Options | Filter/Highlight (or CTRL-L) and change the * in the Include text box to the name of your process when it runs. Next, launch Regmon and, again, filter the results to include only registry access made by your application (see Figure 4-2). After you have set the filters, clear both windows so you will have clean output.

Next, launch Sysinternals' Process Explorer or your favorite utility (**netstat -ano**) that displays network ports opened by your application. If you expect the application to interact

Vulnerable Installation

There have been famous cases of vulnerabilities introduced by simply installing an application. Microsoft Visio's default installation a few years ago, for example, also installed a version of SQL Server with no password on the all-powerful sa account. Unfortunately, this version of SQL Server (MSDE) also ran as LocalSystem, allowing complete control of the operating system after you log in with the sa account, leading eventually to the SQL Slammer worm. Several database-based web bulletin board systems on Linux have created similar security holes by creating a test user with a default password that is never changed after installation.

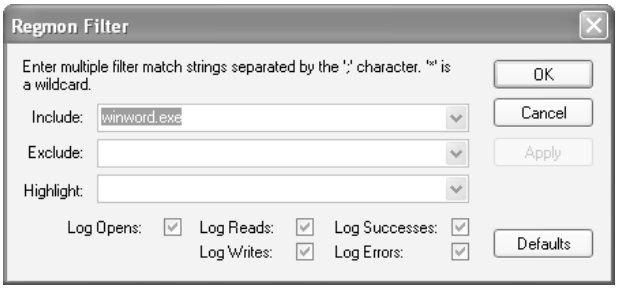

Figure 4-2 Regmon filter window

with the network (perhaps even if you don't expect it), launch your favorite packet sniffer as well. Finally, launch the application. The initial flurry of file and registry access is especially interesting as it retrieves and stores the application state. After the activity levels off, save the Regmon, Filemon, and sniffer output, calling it the initial launch data. Now that the app has been launched once, you can hopefully simulate a normal launch. Run through the sniffer, Filemon, and Regmon routine and launch the application again. Use the application, enticing it into as much network, registry, handle, or file access as possible.

You should be able to collate all the registry keys touched and any file or network access from the logs you've collected so far. At this point, turn away from your data gathering utilities for a moment and take some time to think about any other way the system could be coerced into consuming external data. For example, can your application be customized in any way using any type of configuration file? Most recent MP3 players can be "skinned," meaning a configuration file specifying the look and feel of the player can be specified by the user. If your application can be skinned, add that configuration file to the attack vector list. What do you know about the protocol a connecting network client or server uses to push or pull data to or from your application? You might need to spend some time with a sniffer and a server for your client or a client for your server. Try to outline the protocol during the footprinting phase.

Footprinting Uninstallation Occasionally an application uninstall process will leave behind remnants from the installation. There are only a few cases where this is interesting, but it might be worth your time to uninstall the application on a virtual machine to see if the files and registry keys removed match the files and registry keys created. One

Footprinting with lsof

If you are testing a Linux or Unix-based application, use the open source tool **lsof** to do the same thing as described here for Windows. The name **lsof** stands for LiSt Open Files and it does just that. Its output is a little raw, but it shows all files that are open by processes currently running on the system and also list ports opened by each process. You can download **lsof** at http://freshmeat.net/projects/lsof/.

interesting example of an incomplete uninstallation becoming a security vulnerability was when a vendor recently released a patch for some buggy software with workaround steps to simply uninstall the application. It turns out, however, that the uninstall left the vulnerable web-accessible component in a web-accessible location. (Whoops!) And, of course, the patch didn't update the faulty component because the update software believed the component was uninstalled and removed. (Double whoops!)

Focused Exploitation

Now that you have a nice, long, complete list of attack vectors, go down the list and find a way to break the system. It's hard to give more guidance than that because every system is different. This is where the art form comes in and your creativity tells you where to go. Here are a few exploit techniques you can use for the attack vectors discussed previously. There are lots more, but the important thing is to understand the process and see why the footprinting is so important.

Exploiting Files Parsing files is harder than you might think. It's not so hard to parse a well-formed file, but it is quite difficult to guard against every possible way a file can be malformed. Because of this, many applications do and will continue to break when you substitute malformed data for certain important parts of the file (such as sizes and offsets). Later chapters will talk about passing applications malformed data and how you can construct especially dangerous malformed data.

Exploiting Registry Keys Registry keys are interesting if they live in a registry hive that is writable by unprivileged local users. Narrow your registry attack vectors down to limited-user-writable keys and look for interesting keys. There are vulnerabilities to be found by writing malformed or too much data to unprotected registry keys. It's only marginally interesting to find a set of malformed data in a registry key that crashes the system when only administrators can modify the data. However, when you can find a registry key that any limited user can modify and a blob of data to put into that registry key that crashes the process, you've found an interesting vulnerability. When your footprinting reveals a registry key that is used and weakly protected, use a methodical approach to fill the key with a string of A's, then with binary data, then with numbers of increasing power of two, then with anything else you can think of depending on the data the application is expecting.

Also look for any registry keys that modify the application's launch. For example, perhaps the default command line arguments the application uses when launching are stored in a weakly protected registry key. If you can modify the command line arguments you can tack on an argument to run your arbitrary script by adding a **&& MakeMeAdmin.cmd** to the command line arguments. When the process is next launched, it will also run your MakeMeAdmin.cmd script that you can build to add an administrative user for your use.

Exploiting Named Pipes Named pipes are interesting because the program that listens on a named pipe can impersonate the caller. In plain English, if the application you're testing runs as LocalSystem and connects to a named pipe that an unprivileged user can create, the unprivileged user can run commands as LocalSystem and it's game

over. For those unfamiliar with named pipes, they are a mechanism in both Unix/Linux and Windows that allow processes on the same machine to communicate with each other by sharing a section of memory. Pipes can be anonymous or named. Named pipes are made unique by their name—there can be only one pipe with a certain name on a system. If process A normally connects to process B's named pipe and an attacker can create the named pipe before process B creates it with the same name process B was going to use, process A will connect to the attacker's named pipe, not process B's named pipe. This is a variant of name squatting. On Windows machines, the process that created the named pipe can execute commands with the privilege level of the process connecting to the named pipe. As you can imagine (and has been proven in Microsoft security bulletin MS00-053), when a limited user can create a named pipe that a highly privileged process connects to, you have a bad security vulnerability.

Exploiting Weak ACLs The reason we look for weakly ACL-ed directories is because the location of files can be significant. If you find a privileged application that has allowed its install directory to be world-writable (so anyone can write log files, perhaps), you can own the box. For this trick, simply create an empty file having the same name as the launched executable with ".local" tacked on, such as myapp.exe.local. The presence of this file will tell the application to look in the current directory for the DLLs it needs to load. All DLLs listed in HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs are loaded from system32, but any other DLLs will be loaded from the current directory, giving you a chance to swap the original DLLs out for your malicious custom-built copies of the original DLLs. These DLLs contain the guts of the program and describe what the program does when it is run. If a limited user with some disassembly knowledge (required to build the replacement DLLs and beyond the scope of this book) can take over execution of the program, you should point this out to the customer as a severe vulnerability.

If you can find an application's directory that is weakly protected and also find that the application runs as a service or is launched by an administrative account from a shortcut, it's even easier than building replacement DLLs. In both these cases (launched as a service or from a shortcut), the launch path is hardcoded so you can simply move aside the original file in the weakly protected directory and replace it with your MakeMeAdmin.exe malicious code. For example, let's say you are assessing the security of a service named MyGreatService.exe. The directory \Program Files\MyGreatService\ allows global full control. This allows an unprivileged user to move MyGreatService.exe aside to MyGreatService.bak and copy your malicious program to MyGreatService.exe. The next time the MyGreatService service is started or the next time the administrative user clicks the MyGreatService shortcut, your malicious program will be run instead of MyGreatService, all because of a weak access control list on the program's directory.

Exploiting via the Network This is, of course, the most exciting attack vector because the network stack is accessible remotely and anonymously. We discuss it last, however, because the other attack vectors are also important to check on a thorough system test.

The best way to find vulnerabilities in a network-facing client is to build a malicious server. To assess a network-facing server, build a malicious client. We asked you to outline

the network protocol earlier because you need it now in this step. The process to test the network attack vector is similar to testing files and any other attack vectors that feed the system data. (Feed it some good data, some bad data.) The effort required to build a malicious client is often pretty involved. You first need to understand the protocol, then build a client that negotiates the protocol, and finally corrupt the data you send. However, it is almost always worth the effort. The biggest sources of new security vulnerabilities today are malicious clients/servers and feeding systems malformed data.

Advantages and Disadvantages of System Testing

The best way for someone outside the original developer team to find application vulnerabilities is by focused system testing. Also, this type of testing finds new bugs and doesn't just point out the location of known vulnerabilities. Finally, talented system testers command top dollar on a contract basis.

However, this is the hardest type of assessment to do well. It takes talented, creative assessors to find new bugs in a hardened system. This type of test also requires psychological stamina as the first three weeks of a four-week assessment might find zero security bugs (while you're writing the malicious client) with all the vulnerabilities being found at the very end of the job after you fully understand the application being tested at a very deep level. This might not go over too well with an impatient customer demanding instant results.

References

Greg Hoglund and Gary McGraw, *Exploiting Software: How to Break Code* (Addison-Wesley Professional, 2004).

Michael Howard and David LeBlanc, *Writing Secure Code* (Microsoft Press, 2003).

Reporting Out

After you have gone through the entire assessment thoroughly and found a whole pile of vulnerabilities, the most critical part of the entire process begins. None of your work matters without a well-written report and a final customer meeting that will result in improving their security posture. For the last half of the assessment, your team chief should be working on the report. Given data from the tech lead throughout the process, the team chief needs to build a crisp host-level or vulnerability-level report that you leave with the customer on your last day. It will often take longer to build the high-level, management-style complete summary of your assessment, but you should leave a technical report with the customer so they can get started closing the holes.

Everyone's style is different, but one approach that works well to wrap up an assessment is to schedule three separate meetings for the last day or two of your engagement. First, schedule time with the small group of IT decision makers to share the technical results with them. You want to dig deeply into the vulnerabilities with this group, allowing plenty of time for questions and giving best practices and guidance to fix the holes

that were found. Second, especially for pen-tests and red team engagements, offer to provide training for those who can improve the security of the network. Build a stock security seminar slide deck that your tech lead can customize based on the specific assessment and vulnerabilities found and how to effectively mitigate them.

Finally, schedule a third meeting with the management team, as high up the executive chain as you can schedule. The purpose of this meeting is not to explain the technical guts of your report, because they probably won't understand it. The purpose of this meeting is to demonstrate to the key decision makers in the company why security is important and why they should dedicate people and money to securing their network. You need to have a crisp briefing for this group that includes things that went well (for them), areas they need to work on, and at least one "golden nugget" that gets their attention. Telling a CEO, for example, that you used a DCOM buffer overrun that gave LocalSystem access to a machine that had a domain admin logged on, et cetera, is not going to make an impact. Telling the same CEO that anyone can read his e-mail from the parking lot because of an insecure network configuration will make an impact.

Summary

- The single goal of the pen-test is to compromise the network to point out security gaps.
- Red team engagements include pen-testing but also test the physical security, social engineering, and other aspects of a network's security.
- The system test takes a focused look at a single system or application and finds new security vulnerabilities in its implementation.
- Each team should have a tech lead who directs the workflow of the assessment and is responsible for keeping the quality bar high.
- The team chief interacts with the customer and handles all administrative details of the assessment.
- You should give customers as much detail as they can handle about how you'll be performing your work.
- The phases of a pen-test are discovery, vulnerability enumeration, and vulnerability exploitation.
- **dmitry** is a handy command line tool to pull public information about a network from the Internet.
- During a red team engagement, your team should find as many ways onto the network as possible.
- Red teaming should include an incident response test.
- Enumerating every single attack vector is an important part of an optimal system test.
- More than the network should be footprinted and exploited on a system test.

- Assessments should include three separate wrap-up meetings: one to discuss the results with IT leads, a security seminar to train anyone interested, and a wrap-up meeting with executive management.
- You should try to find a way to make network security "real" for the executive management who may not be as concerned as they should be.

Questions

1. On what type of assessment would you most likely find social engineering attacks?
   - **A.** Pen-test
   - **B.** Red team
   - **C.** System test
   - **D.** Vulnerability assessment
2. Of the following choices, who should be the most experienced, technical hacker?
   - **A.** Team chief
   - **B.** Tech lead
   - **C.** Team member
   - **D.** Customer
3. After the customer has expressed interest in your services, what should be the next contact?
   - **A.** Onsite meeting with customer just before the assessment starts
   - **B.** Letter to customer asking them to sign your "get out of jail free" letter
   - **C.** Assessment planning with the customer, explaining the types of services you can perform
   - **D.** Invoice for a standard pen-test
4. During the kickoff meeting, which is most likely to be discussed?
   - **A.** Buffer overrun vulnerabilities
   - **B.** Recent attacks against organizations similar to theirs
   - **C.** What the assessment team plans to do/accomplish
   - **D.** Likely follow-up actions from the assessment
5. What should be the first step in a pen-test?
   - **A.** TCP and UDP port scan
   - **B.** Alive scan
   - **C.** Open source research into information publicly available on Google, Netcraft, etc.
   - **D.** War driving

6. Which is the best type of test to find new vulnerabilities in a recently deployed application?
   A. Pen-test
   B. Red team
   C. System test
   D. Ad hoc testing
7. Red teaming is best at simulating which threat?
   A. Insider threat
   B. Script kiddie
   C. Automated attack from the Internet
   D. Focused hacker attack from the Internet
8. Which of these utilities comes from Sysinternals?
   A. THC Scan
   B. **dmitry**
   C. Filemon
   D. None of these utilities comes from Sysinternals

Answers

1. **B.** While social engineering could be incorporated into each of the other tests, it is featured on red team engagements. A is incorrect because pen-tests primarily deal with network bits only. C is incorrect because system testing is focused on a single application or system, and D is incorrect because vulnerability assessments are often done by an automated tool not capable of social engineering.
2. **B**. Good tech leads are hard to come by because they are the elite hackers. A is incorrect because the team chief need only interact with the customer and generate the report, at a minimum. C and D are wrong because while it's great to have experienced team members and fantastic to have a security-savvy customer, the tech lead drives the assessment and needs to be the most technical and talented hacker.
3. **C**. It's ideal to go back and forth with the customer multiple times before the team actually starts the test. Focusing the assessment to include those items the customer especially wants included makes for happier customers. A, B, and D are wrong because those other customer interactions must be customized based on your understanding of exactly what the customer wants.
4. **C.** The kickoff meeting is primarily to tell the customer what you will be doing. This is also another chance for the customer to refocus your efforts. The most important takeaway from the kickoff meeting is a team focused on the customer's priorities. A and B are incorrect because those general topics won't help focus

your assessment team. D is incorrect because while you might talk about follow-up actions, the primary (and most correct) focus of the kickoff meeting is about the assessment itself, not follow up.

5. **C**. The first step of a pen-test really should be to see what public information is available from Google, whois, and Netcraft before starting the scans. A and B are incorrect because the alive scan and the port scans should come after the open source research. D is not the most correct answer because you'll generally want to know more about the customer's wired network before researching the wireless network.
6. **C.** The system test is the best way to find new bugs. A and B are incorrect because they are good ways to point out the location of known bugs or insecure configurations but not the best way to find new vulnerabilities. D is incorrect because ad hoc testing is rarely the best way to find known or new vulnerabilities.
7. **D.** Red teaming assumes no access is given so it simulates an adversary who is not given any insider access. Therefore, A is not correct. B and C are incorrect because red teaming includes a heavy dose of social engineering, something the script kiddie or automated attack does not provide.
8. **C.** Filemon comes from Sysinternals. Sysinternals also makes Regmon and Process Explorer, in addition to some other great freeware tools. A is incorrect because THC Scan comes from The Hacker's Choice. B is incorrect because **dmitry** was written by James Greig. D is incorrect because Filemon does come from Sysinternals.