Editors: Matt Bishop, bishop@cs.ucdavis.edu
Deborah A. Frincke, deborah.frincke@pnl.gov

About Penetration Testing

MATT BISHOP
University of California, Davis

Students generally learn red teaming, sometimes called *penetration testing* or *ethical hacking*, as "breaking into your own system to see how hard it is to do so." Contrary to this simplistic view, a penetration test requires a detailed analysis of the threats and potential attackers in order to be most valuable.

What is a penetration test?

Suppose you buy an expensive, brand-new car. You don't want a thief to steal it, so you ask your friend, a policewoman who handles car thefts, to help keep your car safe. Yours is a model with which she's unfamiliar, so she wants to examine the car during her lunch hour as if she were going to steal it. That way, she can see what a thief could do.

This is a penetration test. More precisely, it's an analysis of some aspect of a system. It needn't be a computer system—indeed, it could be a secured building or, more likely, a combination of people, an office, and a computer system. The "aspect" being tested, however, must be stated clearly, which means that the sponsors commissioning the test must tell the testers what they expect them to do. A generic goal such as "break into the system" isn't clear enough: is the goal to establish whether there's at least one way to break into the system, or do the sponsors want to know how easy (or hard) breaking into the system is? For that matter, what does "break into the system" mean—read a confidential file, execute a specific program (or any program), impersonate a user, change data stored on the system, or change the system's behavior by reconfiguring it?

Going back to our example, you need to tell your friend what you mean by "steal." By definition, that means an unknown person drives the car away and keeps it, but is it stealing if the person returns the car the next day? If your newly licensed teenager borrows the keys without your permission and takes the car for a drive, is that stealing? If a transient sits in the car without driving it, is that stealing?

Let's say that you decide the following:

- You care about someone unknown driving away with the car; if they later return it, it's still stolen.
- You'll certainly be unhappy if your teenager takes the car without permission, but you won't report it a theft.
- You might have to call the police to evict the transient sitting in your car, but the vehicle is still there, so it's not stolen.

Now, your friend has a specific goal for her testing: can someone who doesn't have a key drive your car away?

The goal needn't be as absolute as preventing the theft. It might simply be to limit a would-be thief's abilities—requiring more than three hours to steal the car, for example. The idea behind such a goal is that someone would see the attempted theft. An electronic analogue would be to inhibit an attacker who breaks into a system from degrading performance beyond a certain point (*intrusion tolerance*).

What do the attackers know?

After selecting a specific goal (or set of goals), the sponsors must determine the threats against the system. Threats come from adversaries, whose capabilities speak not only to the policies and procedures protecting the system (we'll return to this later) but also to the assumptions that the testers must make. The more information an attacker knows or can determine, the more the testers must also know in order to make the results meaningful.

The "can determine" part is important. Typically, sponsors need reports of system problems and have limited time and resources for penetration tests. Indeed, if the benefits of compromising the system are great enough, attackers will have more resources than testers. To adequately play the attackers' role, testers must thus compensate for the disparity in resources. The simplest method is to ensure that the testers know everything about the system, so that they can take into account everything the attackers could uncover.

Let's return to our car example. What information should you give your friend, the police officer? If you simply show her the car, she'll have to figure out details such as how the door lock is linked to the door latch and how the ignition is wired. This will take time—perhaps more than she has available. If you don't give her the owner's manual or mechanic's guide for the car, she won't be able to evaluate how easy it is to steal the car within a short time.

Moreover, if you lock the car without giving your friend the key, she might spend significant time just trying to get in, as opposed to examining the steering column to see if a thief could hotwire the car. Unlike a thief, who might simply smash the window or pry the door open with a crowbar rather than waste time, your friend won't want to damage your car. To determine, realistically, how hard it is to steal the car, your tester needs access to it.

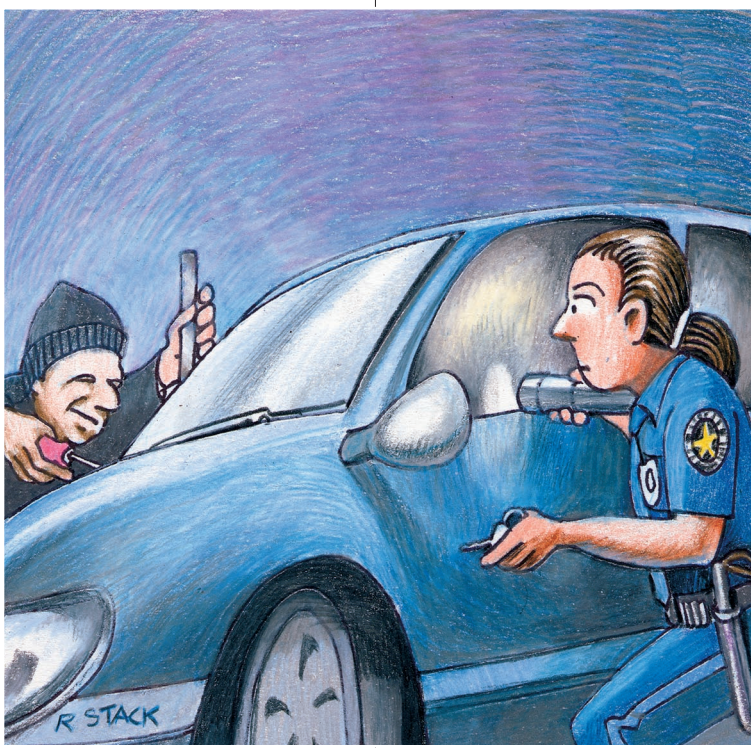
Having students consider whether to give the tester a key is a good exercise. It also illustrates how penetration testing requires thinking that's "outside the box." Many students will focus on the difficulty of unlocking the car, arguing that if the tester couldn't succeed, the thief would also find that doing so was difficult. This is probably correct, but the point is that the thief would likely just bypass that problem altogether.

This answers the question of what testers should know. Attackers might not have as much knowledge, but they usually have more time than the testers in which to acquire it; they might also be able to bypass the need for this information in ways that testers can't. Unless a specific reason ensures that attackers won't know something, testers need to know it.

This leads to the issue of *security through obscurity*, which means the defense is predicated on the attackers' lack of information. Contrary to widespread opinion, this defense is valid, providing that it's used with several defensive mechanisms ("defense in depth"). In this way, the attacker must still overcome other defenses after discovering the information. That said, the conventional wisdom is correct in that hiding information should never be the only defensive mechanism.

When setting up a penetration test, the question of how much information to tell testers depends on what the sponsors want to find out. If the goal is to determine how difficult discovering the information is, testers should get no information, but such a goal is usually inappropriate in this age of widespread Internet connectivity and information dissemination. Not only can adversaries often assemble information from unlikely sources, the protectors of such information often leak it.[1] For example, lawyers for the DVD Copyright Control Association accidentally made public the source code of the encryption algorithm for DVD disks.[2] In a similar vein, government contractors accidentally posted information endangering US soldiers to file servers that anyone could access online.[3] If the goal is to determine how secure a system is, testers should have complete information about it, so that they can determine how effective the controls are against informed and knowledgeable attackers.

Typically, attackers with different skills and knowledge will threaten a system. For example, the neighborhood "thug" might be interested in your new car for a joyride but will have little skill at stealing cars. A professional thief might want to steal the car to sell it, and will likely be very skillful at doing so and (presumably) evading law enforcement. The finance company might need to repossess the car if you don't make payments; their agents will also be skillful, but operate with the support of law enforcement. Finally, a mechanic from the store that sold you the car might want to steal it; although operating outside the law, this person could (presumably) ac-



quire a key with ease and already knows how your car works. This leads to our next question.

What resources do the attackers have?

Once the sponsors determine the set of attackers that are of concern, they must next define the testing constraints, particularly regarding how much time and what resources to give the testers.

Returning to our example, the resources available to each of the four classes of attackers differ. If the goal is to deal only with one class—for example, the joyrider—consider whether testers need the key. Logic says not, because if you always lock your car, the joyrider won't be able to get in. The catch is whether you can be sure that you'll never forget to lock the car. Given that humans make mistakes, the testing should ask not only how hard it is for someone without a key to get into the locked car but also what happens if you forget to lock it.

On the other hand, if the goal is to handle all classes, testers must assume that resources are available to all attackers, or else figure out how to deny them some of the necessary resources. The latter approach might be possible in specific circumstances. Suppose, for example, the computer system under test requires that users have physical access. One approach to securing the system is thus to deny all attackers physical access to it. Again, however, this approach requires that the mechanisms enforcing the denial work perfectly. As in the case of the joyrider, testers should ask what happens if the controls fail.

This raises an additional issue of interpretation. When presenting results, testers shouldn't simply say whether the system is vulnerable. Instead, they should describe the specific vulnerabilities they've found as well as how those flaws can be exploited. From this, the sponsors (or testers) can determine which attackers threaten to exploit particular vulnerabilities. As an example, if a vulnerability in the car's electronics that requires specialized knowledge to exploit lets an attacker open the car door without a key, the joyrider will be unable to exploit that particular vulnerability. But the mechanic or professional car thief might be able to do so because they know about those electronics.

The advantage of such a detailed breakdown of vulnerabilities is that sponsors can evaluate new threats in light of what testers have found already. Thus, testing results become useful in determining how security mechanisms handle new threats.

Application

In an earlier column, we discussed security-relevant requirements for electronic voting.[4] Consider the goals of a penetration study for an e-voting system. What might some goals of this study be?

1. Determine whether someone could change the votes recorded on the system.
2. Determine whether someone could alter the firmware on the system to display votes correctly but store them incorrectly.
3. Determine whether specific policies and procedures sufficiently prevent attackers from changing the results.

These goals overlap, and the specific goal (or set of goals) depends on what the study aims to show. For example, goals 1 and 2 were part of the penetration exercise (although expressed in a different form[1]) in the red teaming for a top-to-bottom review of e-voting systems in California. Goal 3 was an aim of the overall review, but not the penetration study. To satisfy it, the red teams reported the specific technical vulnerabilities to the California Secretary of State, who determined whether election officials were using policies and procedures that were sufficient to counteract the threats.

Several types of attackers exist for voting systems:

- the voters, who use the systems to cast votes;
- the poll workers, who set up and tear down the systems on election day;
- the election officials, who use the systems to count and report votes;
- the technicians, who fix and maintain the systems; and
- the vendors, who write the software for the systems and design and implement them.

A good classroom exercise is to examine some previously located vulnerabilities and ask who could exploit them. Consider a vulnerability in the cryptography that signs a memory card once the polls are closed; at that point, the card shouldn't change, but which of these classes of attackers could alter the card? Which could do so without detection? The answer depends on the system's design and implementation details, as well as the implementation of the procedures controlling access to the cards.

Ethics

Teaching penetration testing requires that students learn to appreciate the ethical issues involved. Penetration testing lets the teacher discuss and bring to life various ethical codes and forces students to think about the consequences of their actions. At the very least, the penetration test's parameters should be in writing so that testers and sponsors agree on the testing's goals and limits. For example, is social engineering allowed? Can testers use systems other than those they're testing (to crack passwords)? Written agreement on testing parameters protects everybody.

Using the results of penetration testing requires proper interpretation. Neither testers nor sponsors should assert that the penetration test has found all possible flaws, or that the failure to find flaws means that the system is secure. All types of testing can show only the presence of flaws and never the absence of them. The best that testers can say is that the specific flaws they looked for and failed to find aren't present; this can give some idea of the overall security of the system's design and implementation.

Penetration testing is effective because it lets us consider a system as it's actually used, rather than as it's expected to be used. It's something to which all computer security students should be exposed. □

References

1. M. Bishop, *Overview of Red Team Reports*, tech. report, Office of the Secretary of State of California, 2007.
2. M. Bishop, *Computer Security: Art and Science*, Addison-Wesley, 2003.
3. "Government Agencies Posting Sensitive 'Need to Know' Material Online," *Fox News*, 12 July 2007; www.foxnews.com/story/0,2933,289011,00.html.
4. M. Bishop and D. Frincke, "Achieving Learning Objectives through E-Voting Case Studies," *IEEE Security & Privacy* vol. 5, no. 1, 2007, pp 53–56.

Matt Bishop *is a professor in the Department of Computer Science at the University of California, Davis, and a codirector of the Computer Security Laboratory there. His research interests include vulnerabilities analysis, the design of secure systems and software, network security, formal models of access control, and intrusion detection. He is the author of* Computer Security: Art and Science *(Addison-Wesley, 2002). Contact him at bishop@cs.ucdavis.edu.*