Term Project

For this term you will perform a vulnerabilities analysis of a business company, Giving Announcements and Grading Assignments, LLC (GAGA). This company provides a web-based service for announcing and grading homework assignments, and uses a web server to provide the announcements (which include the assignments and their due dates). Grades are kept in a set of files on the computer, and can be accessed again via the web. A server hosts the web server.

The recent attacks on Spamhaus, and on the New York Times, have made GAGA aware that connecting to the Internet for their business may pose some risk. This risk might be amplified, they feel, by having their web and grading servers accessible to the world. So they have asked whether a penetration test can help them be sure their systems are secure, and if so what is the business case for such a test.

The company you work for, Penetration Testing and Assessments (PTA), will work with business people to develop this. Your specific jobs are to carry out the vulnerability testing and report results to your business colleagues, who will use the results to make their case.

GAGA's immediate concerns seem to be the following:

1. Can an (authorized or unauthorized) user of the grading system change the due dates on the assignments?
2. Can an (authorized or unauthorized) user of the grading system change his or her grades?
3. Can an (authorized or unauthorized) user of the grading system see or change another student's grades?
4. Can an (authorized or unauthorized) user of the grading system acquire *root* privileges?

But, as they are new to the Internet, they are not sure these are the only threats they should be worried about.

The server is the host *noevalley.cs.ucdavis.edu* (IP address 169.237.6.201). It is running a UNIX-like operating system called "FreeBSD 9.1-RELEASE)" — if you are used to Linux, it is very similar. Source code is available in the directories */usr/src* (the source code for the core system) and */usr/ports* (the sources for various other programs).

The ground rules are these: you may *not* use a denial of service attack from the network or on the system. You also may not tamper with other users' files. You *may* alter system configuration files or files in the web server area (*/usr/local/www/data*), if you can get to them in that way. The homework files are all in that area, under the subdirectory *classes*. Grades are kept in that area also, under the subdirectory *grades*.

As soon as you form your team, please submit the names of the members of the team (including your own) to SmartSite, in the "Project Teams" assignment area. I will put an account name and password for the system into the "Instructor Summary Comments" area. You will have to use *ssh*(1) to log into the system, of course.

Project Organization

You will work in groups of 3 or 4. Each group will work with students in an information security management course at the School of Business; when you get your login and password, I will also give you the email address for your counterparts (it will be on *noevalley*). Note: please send mail to these addresses only when you are logged into *noevalley*. Otherwise, it will go through the UC Davis main server, and bounce. And beware — this school is in Perth, Western Australia, which is 15 hours ahead of us! This means your communications will have to take into account this difference.

You and your colleagues are responsible for determining the objectives for the test. The testing will use an approach called the Penetration Testing Execution Standard. This approach is an elaboration of the Flaw Hypothesis Methodology. It consists of 7 phases:

1. Pre-engagement interactions;
2. Intelligence gathering;
3. Threat modelling;
4. Vulnerability analysis;
5. Exploitation;
6. Post exploitation; and
7. Reporting.

From your point of view, the key ones are:

Intelligence gathering and *Threat modeling*. In these, you will gather information about the target. You can use tools like *nmap*(1) to determine which ports are open, and simply poke around the system to look for configuration and file permission issues (and other things). Source code is available, as above, if you want to examine that. Use this step to gather information and hypothesize flaws, bearing in mind the objectives. Please keep a record of each flaw you

believe may exist, and your basis for that belief. Be specific here: saying "this is a FreeBSD system, so it might have the vulnerability XYZ" is not sufficient. But saying "this is a FreeBSD system, port 139 is open, and the server on that port is known to have vulnerability XYZ" is. In other words, you mist find something specific to the system you are analyzing in order to hypothesize the flaw. At this stage, do *not* test your hypothesis — that comes later.

As you develop your hypotheses, send them to your business colleagues. He or she may also suggest things to you that you should evaluate, or point out consequences if that vulnerability actually exists (sometimes of the effects it may have). Your colleague will look at these hypotheses and decide which ones pose the greatest threat to the client's environment. You and he or she must also decide which ones have the most likelihood of success. The result of this step will be an ordered list of vulnerabilities to be checked.

You do *not* have to list all the hypotheses to begin checking them. In other words, once you have hypotheses you feel are worth checking, begin checking them. When you hear from your colleagues, you can then order them and proceed down the list. Also, you do not have to check them all.

For this project, you must check at least 3 potential vulnerabilities, and document them as described below.

Vulnerability Analysis and *Exploitation* Now check them. Try to confirm whether the vulnerability is present *without* exploiting it if you can; this is much more challenging, because you have to think of a way to demonstrate it. But if you need to, try to develop exploits for the vulnerabilities you think may be there. You will work with your colleagues to plan how to exploit the vulnerability to achieve the needed access. As before, please keep a log of what you try, and what happened. Include output from any tools you use here, as well as identifying the tool (or submit a copy of it with your final write-up). We should be able to reproduce your results, so please ensure your description is detailed enough for us to do so. ***Include exploit attempts that fail here — those are as significant as those that work!***

Post exploitation Here, your business colleague will lead the way; you are to help him or her achieve the goals they need. This could involve copying files, inserting new files, deleting existing ones, or inserting Trojan horses (usually back doors) to allow an attacker to enter the system with minimal fuss. When you do this, your goal is avoid detection. And no, we won't tell you whether, or how, we will be monitoring the systems to detect you! But don't forget to clean up after yourself.

When you do this, be sure you record the session, so we can see what you do. This is also for your protection, because if GAGA claims you have damaged their systems in a way that is not allowed, the session recording will show them you did not.

Reporting Your job here is to present three write-ups describing the vulnerabilities you tested for, including a detailed description of why you think the vulnerability might have been there, how you tested for it, and the results of the test. Please use the structure in the Appendix for your write-up. You are *not* to make a business case for or against penetration testing; that is your colleagues' job.

What Is Due and When

Please submit the following on the dates indicated:
Project group: due on Monday, May 20; 10% of project score. Submit the names of your team members to the project part 1 area of SmartSite.
Completed project: due on Thursday, June 6; 90% of project score. Turn in the list of vulnerabilities you brainstormed in the intelligence gathering and threat modeling steps, and your write-ups for at least 3 vulnerabilities, along with any supporting material (screen shots, tools you used, etc.) If the tool is one you wrote, please include the source code; otherwise, simply give the URL or source of the tool. Be sure you have enough detail in the write-up so we can duplicate your test.

As I mentioned in class, grading will be on creativity and originality of your hypotheses (both in the list and in the write-ups), how well your tests test for the vulnerabilities, and how thorough your documentation is. It will *not* be on whether or not you succeed in compromising the system — but if you do and meet other requirements, that's definitely a plus!
In all cases, submit the project to SmartSite as described in **All About Homework**. If a team has multiple members, only one need submit the material, and the others can simply submit a note saying who submitted the final project.

Vulnerability Write-Up Form

Background

Put enough information in here so that the vulnerability you will be considering makes sense. You may assume that the reader knows basic Linux/UNIX commands and ideas, but anything beyond that should be documented.

Vulnerability Hypothesis

State the specific vulnerability, making reference to the background material when necessary. Explain both what the vulnerability is, and what happens when it is exploited — what changes in the system (for example, a process increases its privileges) and how that gets you closer ti the goals outlined above.

Vulnerability Test

Design and describe a test that will determine if the vulnerability is exploitable. This may be source code analysis, a tool to try to exploit the vulnerability, or something else. Describe the test in enough detail that you could give your description to someone else and they could carry out the test without any further communication between you two.

Vulnerability Test Results

Run the test and show the results. **A failure is as good as a success, provided the test is carried out properly.** Include any screen shots showing the relevant output, and annotate everything so that what happened is clear.

Vulnerability Generalization

If the vulnerability is an exploitable one, try to think of other vulnerabilities that might result from this one being successfully exploited. Then add them to your list to be tested!