CS 6262: Network Security April 15, 2009

Problem Set 3: Wireless Security and Buffer Overflows

Instructor: Prof. Nick Feamster *College of Computing, Georgia Tech*

This problem set has two questions, each with several parts. This is not a coding-intensive assignment; rather, we want you to learn about the tools and techniques to exploit certain vulnerabilities. The report is the most important part of your submission, so please be as clear and concise as possible.

You may discuss ideas with others in the class, but your solutions and presentation must be your own. Do not look at anyone else's solutions or copy them from anywhere. Please cite all resources (web or otherwise) you used to arrive at your solution. (Please refer to the Georgia Tech honor code, posted on the course Web site.)

Turn in your solutions in on **April 24, 2009** by 11:59 p.m. -0500, to the grader, Priyank Raj (priyank.raj at gatech.edu).

1 Wireless Security

1. Many people believe that MAC address filtering is sufficient for home router security. Are they right? If not, how would you break into a router with MAC filtering? In what cases will your attack not work?

2. Briefly describe some recent attacks against WEP? Specifically, skim through the following papers:

   - A. Stubblefield et. al, "Using the Fluhrer, Mantin and Shamir Attack to break WEP", NDSS 2002.
   - A. Bittau et. al, "The Final Nail in WEP's Coffin", IEEE Symposium on Security and Privacy 2006.

   Do you think WEP would be secure if it used a secure stream cipher algorithm instead of RC4?

3. Download the WEP-cracking suite Aircrack-ng from `http://www.aircrack-ng.org`. How would you crack a WEP key (on a network with a visible SSID) using this suite? Please provide the details of the attack and the time taken to crack an SSID (you can use your own home network). Can you crack a WEP-secured network with a hidden SSID (e.g., GTwireless) using this tool? If so, how? (Feel free to include screenshots in your report.)

2 Buffer Overflows

In this problem, you will exploit an HTTP server program that has a stack overflow vulnerability. Your task is to create a *remote* buffer overflow exploit that gives you control over the system running the server program. You will be working only with the binary of the server program, but discovering and exploiting vulnerabilities in such programs is very much possible.

1. Download the server program for your platform from `http://davis.gtnoise.net/~avr/cs6262/ps3`. Make sure it runs, and familiarize yourself with the usage for the program. The server program supports downloading files over HTTP 1.0. If you need code for another platform, please email the TA.

2. Next, you must discover the vulnerability. Because there is just one user-provided string (the HTTP request), you need to figure out the length of the buffer that will overwrite the return address. Please refer to Aleph One's excellent *Smashing The Stack For Fun And Profit* article to get started. You will also need gdb to trace the execution of the server. (Hint: the buffer you need to overflow is less than 300 bytes.)

3. Once you figure out the length of the request string that overwrites the return address, you must overwrite the return address to jump to an address *within* the string that contains the exploit shellcode. Newer Linux distributions by default randomize a process's stack, which makes it tougher to guess the new jump address. You will need to disable this to perform the exploit. On Ubuntu or Debian, do (as root)

   ```
   /sbin/sysctl -w kernel.randomize_va_space=0
   ```

   Some distributions (e.g., Redhat) have ExecShield on; to disable that, do

   ```
   /sbin/sysctl -w kernel.exec-shield-randomize=0
   ```

   You can verify that stack randomization is disabled from gdb by viewing the registers at some point in the program; the values should be the same across runs.

4. Finally, you must exploit the overflow using shellcode that performs some form of privilege escalation. Remote buffer overflow exploits can do many things: read a system file (e.g., /etc/passwd), bind to a local port and connect it to a shell (so-called "bindshell" attacks), connect a local shell to the attacker's machine ("reverse shell"), etc; you are free to choose any of these ways. You can find example shellcode at `http://milw0rm.com` and `http://www.shell-storm.org/shellcode/`. (*Note: not all shellcodes you find on the net are trustworthy. Please do not execute any of these on your machine as a privileged user; we suggest you create a new unprivileged user or run everything inside a virtual machine.*)

In your report, describe your observations and progress at each step. Also provide your exploit as a program (script or C source) which we can run by just changing the value for the new return address.