Network Security Tools is divided into two sections. The first section is “Modifying and Hacking Security Tools” and includes the first six chapters. The remainder of the book is “Writing Network Security Tools” and includes five more chapters. The intended audience is Linux or Unix users but the author goes to great lengths to include helpful information for Windows users. Tools described in this book were all “Open Source” projects. The Authors expect people to be familiar with C and Perl. Extensive code examples were included as well as a website reference to download examples from the publisher.

My impression of the “Network Security Tools” could be assessed as: #1 - How to find vulnerabilities. #2 - How to best exploit these vulnerabilities. I would recommend this book as a must read for anyone who is really interested in becoming a hacker! Having said this, it can be very beneficial to test your own equipment against these standard scanning tools to understand your own vulnerability. I don’t believe that anyone really needs to build a “proof of concept” virus. Although some will argue that it strengthens code by forcing manufacturers to improve their code. Also, hopefully everyone already knows that FTP credentials are not secure! We all like to think that other credentials are secure though.

Chapters:
1. Writing plug-ins for Nessus
   Nessus is a standard vulnerability scanner, with a “NASL” scripting language. Nessus has over 8,000 plugins, and is in use by over 75,000 organizations worldwide. A quick reference for the NASL Interpreter is provided. Syntax and flow control are explained, as well as function definition. You are presented with 30 pages dedicated to programming the interpreter. A plethora of examples are provided to help explain Nessus capabilities.

2. Developing Dissectors and Plug-ins for the Ettercap Network Sniffer
   Ettercap is a network sniffer, capable of dissection of protocols and logging network traffic. Traffic dissection could be useful to pull names and passwords out of the network traffic. Ettercap can also issue alerts when specific traffic is recognized on your network. The dissector example code captures username and password on FTP traffic.

3. Extending Hydra and Nmap
   Hydra can be used to crack logins utilizing many different protocols. Hydra is not extensible, or doesn’t have a macro language. But it can be re-compiled to include new “c” code to define a new protocol. The source code for an SMTP authentication tools is covered in this chapter. Nmap should probably be a separate chapter, but the
information available was fairly brief. Nmap is a port scanner, and can be extended. By default, if Nmap can’t recognize a port response, a fingerprint is generated to help detect it in the future. To extend the match list, make sure you are ready for Perl and regular expressions!

4. Writing Plug-ins for the Nikto Vulnerability Scanner
   Nikto is a highly recommended web server scanner. It scans over 3200 potentially dangerous files/CGIs on over 625 servers. Nikto is also written in Perl. Nikto uses comma separated name.DB files for its data. Core functionality can be extended through the LibWhiskers Perl module. Examples are given to create a brute force attack against NTLM, and also an attack to read mail in Lotus Domino, without authentication.

5. Writing Modules for the Metasploit Framework
   Metasploit is described as a rapid development tool for exploits. It contains an extensible API for setting variable parameters of an exploit. Once again we are looking at a Perl tool, but it has been ported to windows using the Cygwin tools. Metasploit is also usefull for fingerprinting an IDS (Intrusion Detection System) and since the publishing of this book, has been extended to be more stealthy, and not showup plainly in the attacked IDS logs. This chapter also gives a refresher course on how memory buffers operate and how to control the results of an overflow.

6. Extending Code Analysis to the Webroot
   Java code can be checked with PMD code checker. PMD can be extended using a combination of Java and XML code. The rules set from PMD should be considered when you are generating other script code. PHP, Python, Perl, Dot Net tools, ASP… Websites have become the “easier” hacking location. Bug tracking and debugging skills are also discussed here.

7. Fun with Linux Kernel Modules
   This chapter concentrates on hiding processes and intercepting system calls through kernel modules. Can anyone say “Root Kit”?

8. Developing Web Assessment Tools and Scripts
   This chapter contains a brief web overview. Get, Post, as well as SSL are discussed. Perl is used with LWP included to build a log file parser as well as a web-page scanner. Common design errors are checked for, as well as common SQL errors. The Burp proxy server is also utilized in the scanning process.

9. Automated Exploit Tools
   The ethical questions of building exploits was overruled by the desire to verify false positives! How do you really know you have the “vulnerability”, unless you test it with a live exploit? This chapter is an extension of the previous chapters SQL scripted checker, pushing against MS and Oracle SQL servers.
10. Writing Network Sniffers
   This chapter primarily deals with utilizing the “libpcap” c-library to capture network packets. Since libpcap requires root privileges to operate, you are opening yourself up to buffer overflow attacks while using these tools. The discussion migrates from wired to wireless, injecting promiscuous mode and rfmon mode for your wireless adapter. A table listing settings for specific wireless cards is included.

11. Writing Packet-Injection Tools
   Chapter 11 is a direct progression from Chapter 10’s sniffing, utilizing the libnet c-library functions to build custom forged packets. Irregular or “illegal” packets can be utilized to further fingerprint or compromise a target system or IDS. The code for this chapter creates an “I am” executable that creates an ARP poisoning attack. An additional usage would be to combine the libnet’s packet creation functionality with the libpcap sniffer to make a SYN scanner. By forging a tcp/ip connection request, you can determine whether a specific port is open (SYN ACK returned) or not (SYN RST returned), without completing the negotiation. The hope is that without IDS, the reconnaissance is not logged. Also, the “from” address can be randomized to further obscure the attack. Wireless packet injection is also discussed using the “airjack” program.

Reviewed by:
Denton Yoder
Computer Systems Engineer
Biological Systems Engineering
Virginia Tech