Hacking: The Art of Exploitation 2nd Edition

Review by Ryan Linn, CISSP, MCSE, GPEN

Hacking: The Art of Exploitation 2nd Edition (HTAoE) by Jon Erickson is frequently considered a "must read" for those wanting to understand exploits and exploit development. So when I wanted to understand more about the exploit development side of security this was the first book I picked up.

When talking about a book that involves programming, it is often beneficial to know where the reviewer is coming from. I do Windows, Unix, and network security, and I am pretty comfortable with programming although by no means a professional programmer. I have worked some with assembly programming, albeit in the days of Windows for Workgroups, and I really wish that I'd paid better attention in that class in college. Although I do have some experience in these areas, I'm going to point out what areas may cause individuals who haven't been exposed to much programming challenges, and also what areas should be understandable by everyone.

Free Sample Chapter Available Below

"0x300 EXPLOITATION"

del.icio.us

Discuss in Forums {mos_smf_discuss:Book Reviews}

Click Here to download "0x300 EXPLOITATION"

HTAoE is part of the No Starch Press line of books which concentrate on open source, security, hacking, and programming. The book comes with a companion LiveCD which contains Ubuntu and all of the source code and tools used in the book. The chapters are arranged logically starting with programming, then moving to exploits and networking, shellcode, countermeasures, and finally cryptography.

The first sentence or the prefix does a great job summing up what this book is about: "The goal of this book is to share the art of hacking with everyone." This is an honest statement as this book has something for every skill level. When it comes to technical books covering a breadth of knowledge like this one, logically orienting the thoughts in the book is important. When someone has reached his knowledge limits, the book allows the reader to access other resources to augment their knowledge and then proceed. HTAoE begins with a strong introduction bringing the reader into the world of hacking and explaining what hacking is about. It exposes the reader to the terms and the mindset of the culture of exploration and problem solving.

The second chapter is the "Programming" chapter. This chapter transports the reader into the world of programming. The reader is exposed to programming concepts and terms and moves through pseudo-code examples of the basic constructs of programming such as loops, conditional statements, variables, functions, and operators to assure a base level of understanding. Even those who haven't done any programming up until this point should be ok. It is also at this point that the LiveCD will become useful. If you have no Linux exposure, you may want to check out some online tutorials first so that you can be comfortable with Linux syntax, directory navigation, and and the basics of file viewing and editing.

The reader is then immersed in actual code, compiling a sample program, and then looking at an assembly dump of the sample program. The book provides the reader with a brief overview of what is happening with the assembly code and then walks through using a debugger and understanding the x86 registers and assembly basics. Inexperienced programmers may find this section challenging, but sufficient information is presented to enable anyone to find additional specific resources for topics that aren't immediately understood. The programming section moves forward to cover some of the more complex C structures and data types and goes into pointers, typecasting, variable scoping, and understanding strings. The section that follows covers memory and heaps which are extremely important in understanding exploits.

The programming chapter concludes with "Building on Basics" which covers file access and permissions, unix userids, structures, function pointers, and finally looking at and analyzing a game. Most of the applications are short, to the point, and easy to understand. The final application is 400 lines long and detracts from the reader's ability to understand what is going on, especially for those who don't read over code dumps on a regular basis.

Once Erickson has taken the reader through programming basics, the next chapter, "Exploitation," launches right into exploitation of applications. The author explains the concept of why applications are exploitable and includes some real world examples prior to delving into the general concept of exploit techniques. Once the framework has been laid, the reader is thrust into the world of buffer overflows. If you haven't already, I recommend following along with the LiveCD, as this chapter talks about the exploits and heavily uses a debugger to demonstrate what is going on.

The buffer overflow section leads into "Stack-Based Buffer Overflow Vulnerabilities." Here the reader is lead through a number of exercises that illustrate how the buffer overflow can be used to exploit an application. The author uses GDB, The GNU Project Debugger, to demonstrate what is happening with the memory in the applications and how program execution is influenced. The next chapter, "Experimenting with BASH," adds some basic Linux command line tools to the mix in order to facilitate testing vulnerabilities and creating ad-hoc exploits from the command line level. This leads into "Using the Environment," where a demonstration of how to use the environment variables of the shell to hold exploit information allows an exploit to later access this information from within the vulnerable application.

The "Exploitation" chapter progresses to explain other types of exploits such as heap-based overflows, function pointer overflows, and format string vulnerabilities. Each of these are covered in depth with good examples. The 400-line game application pops back up in this section, but the interactive use of it makes what is happening a bit easier to understand. The format string vulnerabilities section is excellent at making a difficult to understand topic more manageable. There are many good examples and walk-throughs using GDB that assist the reader in understanding how to manipulate program memory to control application flow or to reveal information about the application which may not be readily exposed otherwise.

The "Exploitation" chapter concludes with information about some of the lesser covered exploit techniques such as hijacking execution flow using constructors and destructors and manipulating the Global Offset Table. This chapter does a great job of explaining some of the common exploit types. It does not go deeply into Windows, but instead uses Linux and free tools to investigate the basics of some of the most common types of exploits. While this chapter doesn't cover every intricacy, by the end of the chapter, all of the building blocks have been laid for further research, and that appears to be the author's goal.

The "Networking" chapter seemed out of place initially, but after reading the book in its entirety, the networking section is properly placed in order to facilitate the understanding that will be needed in the shellcode section. The networking section assumes the reader possesses some basic understanding of networking topics and begins with some of the more complex subjects like the OSI model. The author then returns to where the programming chapter left off by introducing some of the basic functions required to setup sockets. Erickson builds on these different concepts and works up to writing simple client-server applications. He uses the web as an example and walks the reader through all the steps required to build up and break down network connections. The OSI Model is then discussed in depth, correlating each layer to actual network traffic and then investigating that layer's features and options. The author uses this explanation as a lead-in to "Network Sniffing."

"Network Sniffing" provides detailed information regarding building raw sniffers and pcap sniffers. The author walks the reader through setting up the sockets, capturing the traffic, and looking at the structures involved in building a sniffer to analyze traffic. In this chapter Erickson gives the reader a great example of a sniffer which will break down the packets and print out their hierarchical layers, so that the reader can really visualize what is in the packets. The section progresses into active sniffing, and how to set up ARP poisoning in order to capture specific traffic. The author doesn't just talk about it, but walks the reader through building some of these tools. By the end of the chapter the reader does not just know that it can be done, but he will also understand how this technology works and how to use it. This is a great section for those who are looking to understand how sniffing works, and how to implement a custom sniffer.

"Networking" then addresses denial of service attacks. This section is primarily a discussion of how each type of attack works, but there is some code in the section. There is also information about how one would implement these types of attack. The section serves as a good lead into the later sections that address TCP/IP Hijacking and Port Scanning. These sections have more code and go deeper into how TCP/IP hijacking attacks work. The Port Scanning section mostly talks about the types of scans that can be used and how they would work but uses Nmap for examples.

The final section of the "Networking" chapter returns to hacking, and brings all of the code knowledge from the previous sections of the chapter together to assist the reader in understanding how to create a network based exploit. This chapter contains a good explanation of tracking through the running binary code to follow the attack and determining how to build the exploit. Finally the author walks through the creation of an exploit using port binding shellcode. This final attack brings many of the previous chapters together, and acts as a perfect introduction to the next chapter, "Shellcode."

If you aren't comfortable with assembly code at this point, a stiff drink may be in order. The "Shellcode" chapter has more assembly and debugger output than any chapter. This chapter jumps into investigating the difference between assembly and C and uses the typical "hello world" application for reference. Once the reader has some understanding of what is happening with the shellcode, the author moves into a discussion on how to optimize and sanitize the shellcode for use in exploits. The "Removing Null Bytes" section demonstrates how to use this optimization and sanitation to create functional shellcode without null bytes. The chapter ends by building shellcode for the different tasks that have been exposed as payloads for exploits through the previous chapters. While the review of this section is short, the content is deep and valuable.

The "Countermeasures" chapter focuses on two types of countermeasures: those which detect malicious activity and those which try to prevent malicious activity. Erickson walks the reader through understanding daemonized code and then investigates ways to generate exploits which limit detectability. Included in these methods are socket reuse and payload smuggling as well as creating polymorphic shellcode. Next the author goes into how to overcome stack randomization and non-executable stacks. He presents a sample exploit which exploits a program even though stack randomization is enabled. This chapter is full of code examples that reinforce lessons learned in the programming, networking, and shellcode chapters. This chapter pulls together all the previous chapters, makes use of C, assembly, BASH scripting, Perl, and GDB, and builds applications and exploits that overcome the countermeasures which are most likely to be found in today's systems.

The final chapter, "Cryptology," seems a bit out of place. It only directly builds on the previous programming chapter. This chapter goes over the different types of encryption including RSA, WEP, Crypt, and SSH. Each of the types of cryptography are covered well enough that the reader will know some of the basics, but not deeply enough make him or her an expert. There is a good discussion of man-in-the-middle attacks and WEP attacks including walking through the cracking of a WEP key. There is also a great excercise on setting up a man-in-the-middle ssh hijack. Most of the actual cracking is concentrated on WEP and Crypt with the author building two different types of attack tools for the passwords encrypted with Crypt. Overall, if the reader isn't familiar with encryption techniques, then this chapter will be informative in some areas but possibly too deep in others with the real world examples hitting somewhere right in the middle.

Hacking: The Art of Exploitation 2nd Edition was an excellent read. If you aren't a programmer, this book will give you enough of a taste to have a good basis for doing some additional basic tutorials. As you build your skills, it will have more to offer when you return. By the end of the book you should be comfortable with the basics of writing exploits on Linux even though it does little to touch on Windows. If your only goal is to learn how to exploit vulnerabilities under Windows, this book may not be the one for you. The goals set out in the book were clearly established in the prefix, and I believe that the book delivered.

Ryan Linn, CISSP, MCSE, GPEN - Ryan is currently an Information Security Engineer at SAS Institute. Employed in the computer industry since 1997, he has held positions ranging from web developer to Unix Systems Programmer at a large university to his current position in Information Security. Ryan has been responsible for working with large scale deployments of various flavors of *nix, high availability web and database clusters, as well as for application programming in high availability environments. In the past few years, Ryan has incorporated Windows security into his responsibilities, and is now part of the team responsible for information security globally in one of the largest privately held software companies in the world.