ANALYZING THE STACK-BASED BUFFER OVERFLOW PROBLEM

AHMAD NAZRI BIN ZAINOL

UNIVERSITI UTARA MALAYSIA 2008

ANALYZING THE STACK-BASED BUFFER OVERFLOW PROBLEM

A thesis submitted to the
Division of Applied Sciences, College of Arts and Sciences
in partial fulfillment of the requirements for the degree of
Master of Science (Information and Communication Technology),
Universiti Utara Malaysia

By
Ahmad Nazri Bin Zainol

KOLEJ SASTERA DAN SAINS
(College of Arts and Sciences)
Universiti Utara Malaysia

PERAKUAN KERJA KERTAS PROJEK
(Certificate of Project Paper)

Saya, yang bertandatangan, memperakukan bahawa
(I, the undersigned, certify that)

AHMAD NAZRI ZAINOL
(87998)

calon untuk Ijazah
(candidate for the degree of) **MSc. (Information Communication Technology)**

telah mengemukakan kertas projek yang bertajuk
(has presented his/her project paper of the following title)

ANALYZING THE STACK-BASED BUFFER OVERFLOW PROBLEM

seperti yang tercatat di muka surat tajuk dan kulit kertas projek
(as it appears on the title page and front cover of project paper)

bahawa kertas projek tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan.
(that the project paper acceptable in form and content, and that a satisfactory knowledge of the field is covered by the project paper).

Nama Penyelia Utama
(Name of Main Supervisor): **ASSOC. PROF. HATIM MOHAMED TAHIR**

Tandatangan
(Signature) :

PROF MADYA HATIM MOHAMED TAHIR
Pensyarah
Bidang Sains Gunaan
Kolej Sastera & Sains
Universiti Utara Malaysia

Tarikh
(Date) : 17/11/08

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the university library may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor or, in her absence by the Assistant Vice Chancellor of the College of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

A copy of this thesis may also appear online, on my personal web sites as a very small contribution to the computer security community other than complementing and enhancing my previous, current and future study on computer security.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part should be addressed to:

Assistant Vice Chancellor
College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok
Kedah Darul Aman.

ABSTRAK (BAHASA MELAYU)

Walaupun banyak penyelidikan telah dijalankan selama hampir lebih dari 20 tahun, masalah *buffer overflow* masih lagi berlaku hingga sekarang. Kajian ini mengambil peluang untuk menganalisa *buffer overflow* jenis *stack-based* yang merupakan satu jenis *buffer overflow* yang dominan. Satu demonstrasi eksperimen dalam persekitaran terkawal dijalankan untuk menunjukkan bagaimana dan kenapa masalah ini berlaku. Sepanjang demonstrasi ini, syarat-syarat utama kenapa dan bagaimana eksploitasi ini berlaku dikenalpasti, dianalisa dan didokumenkan. Hasil kajian ringkas ini menunjukkan bahawa terdapat banyak lagi yang boleh dilakukan terutamanya pada peringkat penulisan kod untuk mengelakkan masalah *buffer overflow* ini sebelum kerosakan berlaku, yang selalunya selepas produk perisian telah dijual. Dalam hal ini ianya sudah tentu membazir banyak sumber seperti kos, tenaga manusia dan masa. Hasilnya, beberapa cadangan yang praktikal telah kemukakan di mana ianya sesuai untuk perlaksanaan dan penyelidikan seterusnya. Selain dari itu, hasil kajian ini juga penting untuk dijadikan asas dalam merekabentuk dan melaksana kaedah atau mekanisma baru untuk mengesan dan mengelak masalah *buffer overflow*.

ABSTRACT

It is interesting to know that a buffer overflow problem still exist today despite of many researches have been conducted in a period of more than 20 years. This study takes an opportunity to analyze one of the dominant buffer overflow problem type, a stack-based buffer overflow. A controlled experimental demonstration has been carried out to emulate a stack-based buffer overflow exploit. During the process, main conditions why and how the exploit happens will be identified, analyzed and documented. The findings showed that more works can be done at the coding stage to prevent the problem before the damage (exploit) occurs which normally happen after the software product has been distributed. In this case more resources have been wasted such as cost, man-hour and time. Hence, several practical suggestions with its own advantageous have been highlighted for further research and implementation. In addition, the findings should be very useful inputs in designing and implementing new buffer overflow detection and prevention mechanisms.

ACKNOWLEDGMENTS

Alhamdulillah, because of the blessing and strength from Allah SWT, I am able to complete this project successfully. Shalawat and greeting to our leader, imam, our byword and lover, Prophet of Muhammad SAW (pbuh).

The highest appreciation to my supervisor, Assoc. Professor Hatim Mohamad Tahir for his continuous guidance and feedback. I thank him for the assistance, inspiration and motivation to overcome the difficulties in completing the project.

Next, my greatest gratitude to my online friends around the world that involved in computer security field whom some of them gave me a very useful comments, recommending me for good books and articles. Keep up your good works dudes.

Finally, I would like to thank other lecturers mainly evaluators, librarians and course mates who have directly or indirectly given me encouragement, motivation and assistance throughout the journey of completing this writing.

May Allah SWT abundance His blessing and hidayah to all of us, for contributing to and sharing with others, invaluable knowledge. Allahumma amen. Wassalam

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMD	Advance Micro Device
ASLR	Address Space Layout Randomization
CERT	Computer Emergency Response Team
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
DEP	Data Execution Prevention
DLL	Dynamic Link Library
EBP/ebp	extended base pointer
EIP/eip	extended instruction pointer
ESP/esp	extended stack pointer
GCC	GNU Compiler Collection
gdb	GNU Debugger
GOT	Global Offset Table
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IOS	Internetwork Operating System
ISO	International Organization for Standardization
NOEXEC	No Execute
NOP	No Operation
NX	No Execute
OS	Operating System
POC	Proof-Of-Concept
SEH	Structured Exception Handling
SP	Service Pack
SQL	Structured Query Language
SSP	Stack-Smashing Protector
XD	Execute Disable

LIST OF APPENDICES

LIST OF CODE LISTING

CHAPTER ONE
INTRODUCTION

Almost 20 years after the first publicized buffer overflow vulnerability and exploit used by Morris worm in November, 1988 [1] then followed by Code Red worm in July 2001 [2] and Slammer worm [3] in January 2003, the buffer overflow still one of the top vulnerability, proven to have a severe effect, which has been exploited successfully. Since then, the buffer overflow vulnerable exploited again and again which cover a wide range of computer application, library, operating system (kernel and embedded system as well) and networking. There are many hardware and software based techniques and tools that have been proposed and developed to detect and protect from buffer overflow vulnerable. However this vulnerable still happen and based on the trend it look likes this problem will continue to happen.

It is publicly known that the buffer overflow happens when there is no bound checking on the used buffer in programs. In the plain source codes, this no bound checking is a normal for unsafe programming language that dominated by C and C++ for a set of the standard library functions. Unfortunately, C and C++ are the languages that most widely used for critical applications such as kernel, Operating System (OS), database engine and device driver.

A buffer is small and reusable temporary data storage during the program execution. Normally it is declared as sized array data type in C and C++ programs though it is not limited to the declared sized array because other unsafe standard C and C++ functions used for string and character manipulation such as `strcat()` and `gets()` also resembled similar characteristics. Without bound checking, input size that bigger than the declared size of the array will overwrite other adjacent data in memory and corrupting them. From the programmer point of view, the problem is, when we declared a sized array in a program, we normally already expected or pre-calculated the maximum number of input. However, we cannot accurately determine the maximum number of input from user and other application

The contents of the thesis is for internal user only

Hopefully, by having a better understanding on how and why the stack-based buffer overflow vulnerability and exploit, programmer that using C or other similar 'unsafe language' can avoid this problem at the earliest stage of their task in developing a program. Having a good knowledge where the buffer overflow vulnerability is possible to happen in the application development for example, will obviously contribute something that can improve the product's quality, saving cost, man-hour and time. This is also supposed to be beneficial for other languages that use C as their base code so that the buffer overflow problem does not inherited and propagated.

5.2 Related Future Work

The information provided in this part is actually extracted from the Finding and Discussion section. One future research that can be done is to find the relationship between the program size and speed when we add extra code for buffer overflow protection. This should be specific to buffer overflow codes and should be critical for large program. The effectiveness of the prior secure coding knowledge or training also could be measured when the topics of secure C/C++ coding included in C/C++ syllabus or suitable secure coding training is conducted. This also can be applied when implementing a comprehensive exception handling in C/C++ programming.

Another interesting thing to explore is to enhance the C/C++ editor or compiler with educational info of the buffer overflow problem such as through the intellisense feature. Research can be done to gauge the effectiveness of the implementation on reducing the buffer overflow problem.

This project does not emphasize on the compile and runtime detection and prevention solution other than implementing a comprehensive exception handling because of the many researches (mostly funded) have been carried out as discussed in the Literature Review section. Although it is out of programmer's control, this does not mean it is not important. For example, the convention used for C function call, the way how the stack frame constructed and destroyed (also how the processor support the mechanism through its execution environment) may be reviewed and changed to the better and safer ways.

REFERENCE

[1] Eugene H. Spafford, "The Internet Worm Program: An Analysis," Purdue Technical Report CSD-TR-823, December 8, 1988. [Online]. Available: http://homes.cerias.purdue.edu/~spaf/tech-reps/823.pdf. [Accessed: July. 3, 2008].

[2] CERT Incident Note IN-2001-08, "Exploited buffer overflow vulnerability in IIS Indexing Service DLL," July. 19, 2001. [Online]. Available: http://www.cert.org/advisories/CA-2001-19.html. [Accessed: July 3, 2008].

[3] Gregory Travis, Ed Balas, David Ripley and Steven Wallace, "Analysis of the SQL Slammer worm and its effects on Indiana University and related institutions," Advanced Network Management Lab, Cybersecurity Initiative, Indiana University, May 17, 2007. [Online]. Available: http://paintsquirrel.ucs.indiana.edu/pdf/SLAMMER.pdf. [Accessed: July 3, 2008].

[4] Kelly Jackson Higgins, "Buffer Overflows Are Top Threat, Report Says," Darkreading, Nov. 26, 2007. [Online]. Available: http://www.darkreading.com/document.asp?doc_id=139871. [Accessed: July 4, 2008].

[5] Cisco Systems, Inc., "Cisco 2007 Annual Security Report," Cisco Public Information, 2007. [Online]. Available: http://www.cisco.com/web/CA/events/pdfs/Cisco2007Annual_Security_Report.pdf. [Accessed: July 4, 2008].

[6] Mitre.org, "Vulnerability Type Distributions in CVE," Document version: 1.1, May 22, 2007. [Online]. Available: http://cve.mitre.org/docs/vuln-trends/index.html. [Accessed: July 4, 2008].

[7] Dr. Bjarne Stroustrup, "C++ Applications," Bjarne Stroustrup's personal homepage, July 5, 2008. [Online]. Available: http://www.research.att.com/~bs/applications.html. [Accessed: July 7, 2008].

[8] Mudge, "How to write buffer overflows," insecure.org, Oct. 20, 1995. [Online]. Available: http://insecure.org/stf/mudge_buffer_overflow_tutorial.html. [Accessed: July 7, 2008].

[9] Aleph One, "Smashing the stack for fun and profit," Phrack Magazine, Vol. 7, Issue 49, Aug. 8, 1996. [Online]. Available: http://www.phrack.org/issues.html?id=14&issue=49. [Accessed: July 7, 2008].

[10] M. Kaempf, "Smashing The Heap For Fun And Profit," bughunter.net. [Online]. Available: http://doc.bughunter.net/buffer-overflow/heap-corruption.html. [Accessed: July 7, 2008].

[11] Mingo, "Exec Shield," RedHat people, May 11, 2004. [Online]. Available: http://people.redhat.com/mingo/exec-shield/. [Accessed: July 7, 2008].

[12] Pandey, S.K. Mustafa, K. and Ahson, S.I., “A checklist based approach for the mitigation of buffer Overflow attacks," in *Third International Conference on Wireless Communication and Sensor Networks*, Dec. 2007, pp. 115-117.

[13] Zhenkai Liang and Sekar, R., "Automatic generation of buffer overflow attack signatures: An approach based on program behavior models," in *Proceedings of the 21st Annual Computer Security Applications Conference*, Dec. 2005, pp. 215 - 224.

[14] Smirnov, A. and Tzi-cker Chiueh, "Automatic patch generation for buffer overflow attacks," in *Proceedings of the Third International Symposium on Information Assurance and Security 2007*, Aug. 2007, pp. 165 - 170.

[15] PaX team, "Homepage of The PaX Team," 2002. [Online]. Available: http://pax.grsecurity.net/. [Accessed: July 8, 2008].

[16] Speirs, W.R., "Making the kernel responsible: a new approach to detecting & preventing buffer overflows," in *Proceedings of the Third IEEE International Workshop on Information Assurance*, March 2005, pp. 21-32.

[17] J.P. McGregor, D.K. Karig, Z. Shi, and R.B. Lee., "A processor architecture defense against buffer overflow attacks," in *Proceedings of the IEEE International Conference on Information Technology*, Aug. 2003, pp. 243 – 250.

[18] Nathan Tuck, Brad Calder and George Varghese, "Hardware and binary modification support for code pointer protection from buffer overflow," in *Proceedings of the 37th International Symposium on Microarchitecture (MICRO-37'04)*, Dec. 2004. pp.209 - 220

[19] Zhang Yuhong, Wang Jiebing, Xu Zhihan, Yan Xiaolang and Wang Leyu, "Hardware solution for detection and prevention of buffer overflow attacks," in *Proceedings of 5th International Conference on ASIC*, vol. 2, Oct. 2003, pp. 1304 - 1307.

[20] H. Ozdoganoglu, C. E. Brodley, T. N. Vijaykumar, B. A. Kuperman and A. Jalote, "SmashGuard: A hardware solution to prevent security attacks on the function return address," *IEEE Transactions on Computers*, vol. 55, no. 10, pp. 1271 - 1285, November, 2003.

[21] Zili Shao, Chun Xue, Qingfeng Zhuge, Sha, E.H.M. and Bin Xiao, "Efficient array & pointer bound checking against buffer overflow attacks via hardware/software," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, vol. 1, April 2005, pp. 780 – 785.

[22] Takahiro Shinagawa, "SegmentShield: Exploiting segmentation hardware for protecting against buffer overflow attacks," in *25th IEEE Symposium on Reliable Distributed*, 2006, pp. 277 - 288.

[23] Shao, Z. Zhuge, Q. He, Y. Sha and E.H.-M., "Defending embedded systems against buffer overflow via hardware/software," in *Proceedings of the 19th Annual Computer Security Applications Conference*, Dec. 2003, pp. 352 - 361.

[24] Crispin Cowan, Calton, Dave Maier, Heather, Jonathan, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang, "StackGuard: automatic adaptive detection and prevention of buffer-overflow attacks," in *Proceedings of the seventh USENIX security conference*, 1998, pp. 63-78.

[25] Vendicator, "StackShield: A stack smashing technique protection tool for Linux," Jan. 08, 2000. [Online]. Available: http://www.angelfire.com/sk/stackshield/. [Accessed: July 8, 2008].

[26] Microsoft Corporation, "GS (Buffer security check)," Visual C++ Compiler Options, 2008. [Online]. Available: http://msdn.microsoft.com/en-us/library/8dbf701c(VS.80).aspx. [Accessed: July 8, 2008].

[27] Hiroaki Etoh, "GCC extension for protecting applications from stack-smashing attacks," IBM SSP/ProPolice, Aug. 22, 2005. [Online]. Available: http://www.trl.ibm.com/projects/security/ssp/. [Accessed: July 10, 2008].

[28] Mike Frantzen and Mike Shuey, "StackGhost," Purdue University, 2001. [Online]. Available: http://projects.cerias.purdue.edu/stackghost/. [Accessed: July 10, 2008].

[29] Crispin Cowan, Steve Beattie, John Johansen and Perry Wagle, "Pointguard: protecting pointers from buffer overflow vulnerabilities," in *Proceedings of the 12th conference on USENIX Security Symposium*, August 2003, pp. 91–104.

[30] Rinard, M. Cadar, C. Dumitran, D. Roy and D.M. Leu, T., "A dynamic technique for eliminating buffer overflow vulnerabilities (and other memory errors)," in *Proceedings of the 20th Annual Computer Security Applications Conference*, 2004, pp. 82 - 90.

[31] N. Dor, M. Rodeh, and M. Sagiv., "CSSV: Towards a realistic tool for statically detecting all buffer overflows in C," in *Proceedings of the ACM Conference on Programming Language Design and Implementation*, June 2003, vol. 38, no. 5, pp 55 - 167.

[32] Tsai, T. and Singh, N., "Libsafe: Transparent system-wide protection against buffer overflow attacks," in *Proceedings of the International Conference on Dependable Systems and Networks*, 2002, pp. 541.

[33] Nikolai Joukov, Aditya Kashyap, Gopalan Sivathanu, and Erez Zadok, "Kefence: An electric fence for kernel buffers," in *Proceedings of the first ACM International Workshop on Storage Security and Survivability (StorageSS 2005), in conjunction with the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, pp. 37-43, November 2005.

[34] Madan, B.B., Phoha, S. and Trivedi, K.S., "StackOFFence: A technique for defending against buffer overflow attacks," in *Proceedings of the International*

Conference on Information Technology: Coding and Computing, April 2005, vol. 1, pp. 656 - 661.

[35] Zhu, G. and Tyagi, A., "Protection against indirect overflow attacks on pointers," in *Proceedings of the Second IEEE International Information Assurance Workshop*, April 2004, pp. 97- 106.

[36] Nishiyama, H., "SecureC: Control-flow protection against general buffer overflow attack," in *Proceedings of the 29th Annual International Computer Software and Applications Conference*, July 2005, pp. 149 - 155.

[37] Lei Wang, Cordy, J.R. and Dean, T.R., "Enhancing security using legality assertions," in *Proceedings of the 12th Working Conference on Reverse Engineering*, Nov. 2005, pp. 35 - 44.

[38] Benjamin A. Kuperman, Carla E. Brodley, Hilmi Ozdoganoglu, T. N. Vijaykumar and Ankit Jalote, "Detection and prevention of stack buffer overflow attacks," in *Source Communications of the ACM*, Nov. 2005, vol. 48, no. 11, pp. 50 - 56

[39] C. Cowan, P. Wagle, C. Pu, S. Beattie and J. Walpole, "Buffer overflows: attacks and defenses for the vulnerability of the decade," in *Proc. of the DARPA Information Survivability Conference and Expo*, 1999.

[40] Kelly Jackson Higgins, "Four different tricks to bypass StackShield and StackGuard protection," CoreSecurity, 2002. [Online]. Available: http://www.coresecurity.com/files/attachments/Richarte_Stackguard_2002.pdf. [Accessed: July 10, 2008].

[41] Peter Silberman and Richard Johnson, "A comparison of buffer overflow prevention implementations and weaknesses," BlackHat, USA, 2004. [Online]. Available: www.blackhat.com/presentations/bh-usa-04/bh-us-04-silberman/bh-us-04-silberman-paper.pdf. [Accessed: July 10, 2008].

[42] Steven Alexander, "Defeating Compiler-level Buffer Overflow Protection, ";login: The USENIX Magazine, vol. 30, no. 3, June 2005. [Online]. Available: http://www.usenix.org/publications/login/2005-06/pdfs/alexander0506.pdf. [Accessed: July 10, 2008].

[43] Wikipedia, "NX bit," Wikipedia.org, July 1, 2001. [Online]. Available: http://en.wikipedia.org/wiki/NX_bit. [Accessed: July 11, 2008].

[44] Mastropaolo, "Buffer overflow attacks bypassing DEP (NX/XD bits) - part 1: Simple Call," mastropaolo.com, June 4, 2005. [Online]. Available: http://www.mastropaolo.com/2005/06/04/buffer-overflow-attacks-bypassing-dep-nxxd-bits-part-1/. [Accessed: July 11, 2008].

[45] Skape and Skywing, "Bypassing Windows Hardware-enforced Data Execution Prevention," Uninformed Journal, Oct. 2, 2005. [Online]. Available: http://www.uninformed.org/?v=2&a=4. [Accessed: July 11, 2008].

[46] Sebastian Krahmer, "x86-64 buffer overflow exploits and the borrowed code chunks exploitation technique," Suse.de, Sept. 28, 2005. [Online]. Available: http://www.suse.de/~krahmer/no-nx.pdf. [Accessed: July 7, 2008].

[47] ISO/IEC JTC1 SC22 WG14 Committee, "ISO/IEC TR 24731, Extensions to the C Library, Part I: Bounds-checking interfaces," March 2007. [Online]. Available: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1225.pdf. [Accessed: July 12, 2008].

[48] ISO/IEC JTC1 SC22 WG14 Committee, "ISO/IEC PDTR 24731-2, Extensions to the C Library, Part II: Dynamic Allocation Functions, August 2007. [Online]. Available: http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1248.pdf. [Accessed: July 12, 2008].

[49] cert.org, "CERT C Secure Coding Standard," Jul. 16, 2008. [Online]. Available: https://www.securecoding.cert.org/confluence/display/seccode/CERT+C+Secure+Coding+Standard. [Accessed: July 12, 2008].

[50] Yingbo Song, Michael E. Locasto, Angelos Stavrou, Angelos D. Keromytis and Salvatore J. Stolfo, "On the infeasibility of modeling polymorphic shellcode," in *Proceedings of the 14th ACM conference on Computer and Communications security*, 2007, pp. 541 - 551.

[51] K2, "ADMmutate," Security, Jan. 2002. [Online]. Available: http://www.ktwo.ca/security.html. [Accessed: July 12, 2008].

[52] Metasploit LLC, "The Metasploit Project," 2003, [Online]. Available: http://www.metasploit.com. [Accessed: July 13, 2008].

[53] Next Generation Security Technologies, "Polymorphic Shellcodes vs. Application IDSs," Jan. 21, 2002. [Online]. Available: http://www.ngsec.com/docs/polymorphic_shellcodes_vs_app_IDSs.PDF. [Accessed: July 13, 2008].

[54] Pasupulati, A., Coit, J., Levitt, K., Wu, S.F., Li, S.H., Kuo, J.C., Fan, K.P., "Buttercup: on network-based detection of polymorphic buffer overflow vulnerabilities," in *Network Operations and Management Symposium*, April 2004, vol. 1, pp. 235 - 248.

[55] Hsiang-Lun Huang, Tzong-Jye Liu, Kuong-Ho Chen, Chyi-Ren Dow, Lih-Chyau Wuu, "A polymorphic shellcode detection mechanism in the network," in *Proceedings of the 2nd international conference on Scalable information systems*, Article no. 64, Vol. 304, 2007.

[56] Intel.com, "Intel® 64 and IA-32 Architectures Software Developer's Manuals," 2008. [Online]. Available: http://www.intel.com/products/processor/manuals/. [Accessed: Aug. 5, 2008].

[57] Fedora Linux FTP Repository, "Fedora 9 i386 debug info download," *RedHat.com*, 2008. [Online]. Available: ftp://download.fedora.redhat.com/pub/fedora/linux/ releases/9/Everything/i386/debug/. [Accessed: Aug. 5, 2008].

[58] Debian Hardening Wiki, "Using Hardening Options," *Debian.org*, 2008. [Online]. Available: http://wiki.debian.org/. [Accessed: Aug. 5, 2008].

[59] Kerry Thompson, "How to Disable SELinux," 2008. [Online]. Available: http://www.crypt.gen.nz/selinux/disable_selinux.html. [Accessed: Aug. 7, 2008].

[60] Michael L. Scott, *Programming language pragmatic*, 2nd Edition. San Francisco: Morgan Kaufmann Publishers, 2006.

[61] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: principles, techniques, & tools*. 2nd ed. Boston, MA: Pearson Education, 2007.

[62] 1hall, "Introduction to Writing Shellcode," milw0rm.com, April, 2006. [Online]. Available: http://www.milw0rm.com/papers/51. [Accessed: Aug. 11, 2008].

[63] The GNU C Library, "Setuid Program Example," 2008. [Online]. Available: http://www.gnu.org/software/libtool/manual/libc/Setuid-Program-Example.html#Setuid-Program-Example. [Accessed: Aug. 12, 2008].

[64] Adam Shostack, "Home page - Adam Shostack's SETUID man page" 2008. [Online]. Available: http://www.homeport.org/~adam/setuid.7.html. [Accessed: Aug. 20, 2008].

[65] Kevin F. Quinn, "Stack protector, switches and macros," Mail archive of the gcc@gcc.gnu.org mailing list for the GCC project, Dec. 2005. [Online]. Available: http://gcc.gnu.org/ml/gcc/2005-12/msg00216.html. [Accessed: Aug. 21, 2008].

[66] The GNU's GCC, "GCC Command Options, Options That Control Optimization," 2008. [Online]. Available: http://gcc.gnu.org/onlinedocs/gcc-4.1.2/gcc/Optimize-Options.html#Optimize-Options. [Accessed: Aug. 21, 2008].

[67] Fedora Linux FTP Repository, "Fedora 9 i386 Packages download," *RedHat.com*, 2008. [Online]. Available: ftp://download.fedora.redhat.com/pub/fedora/linux/releases/9/ Everything/i386/os/Packages/. [Accessed: Aug. 26, 2008].

[68] The SEED Project, "Instructional Laboratories for Computer Security Education," Syracuse University, 2008. [Online]. Available: http://www.cis.syr.edu/~wedu/seed/. [Accessed: Sept. 9, 2008].

[69] Vim.org, "An improved vi editor version," vim.org, 2008. [Online]. Available: http://www.vim.org/. [Accessed: Sept. 9, 2008].

[70] Ravi Shankar, and Madan Ganesh, "Vim Intellisense," 2008. [Online]. Available: http://insenvim.sourceforge.net/. [Accessed: Sept. 17, 2008].

[71] Microsoft Corp, "Structured Exception Handling," MSDN Library, Sept. 2008. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms680657(VS.85).aspx. [Accessed: Sept. 22, 2008].

[72] Matt Pietrek, "A Crash Course on the Depths of Win32 Structured Exception Handling," *Microsoft Systems Journal*, Jan. 1997 issue. [Online]. Available: http://www.microsoft.com/msj/0197/Exception/Exception.aspx. [Accessed: Sept. 22, 2008].