Virus Analysis: Annotated Bibliography

Md Enamul Karim[1], Prabhat K. Singh[2], Arun Lakhotia[1]
University of Louisiana at Lafayette[1], AVERT, McAfee Inc. Bangalore[2]

The major research works on malware are done at AV companies and they are usually not published. However, Internet is a good source of virus/worm related white papers and there do exist some research papers published at different forums. To understand research trends, we divided the research areas into some categories. These categories are often overlapping. Before listing the bibliography we indexed the list based on these categories.

Theory: [6, 7, 22, 29, 30]

Overview:[11, 16, 28, 75, 92, 95, 98, 113]

Taxonomy: [16, 116]

Vulnerability: [11, 56, 59, 109]

Analysis: [9, 10, 15, 25, 32-34, 54, 57-59, 68, 69, 92, 93, 97, 110, 123]

Static and Dynamic Analysis: [9, 10, 25, 32, 54, 59, 93]

Obfuscation, Polymorphic and Metamorphic: [25, 49, 50, 57, 64, 68, 75, 82, 102, 103, 122, 124, 125]

Detection /Prevention/Disinfect: [1, 9, 10, 14, 19, 21, 35, 37, 39, 44, 46, 48, 51, 61-63, 70, 71, 73, 76, 79, 80, 82, 84, 85, 87, 90, 93, 99, 100, 102, 104, 108, 112, 114, 119-122, 128]

Application of AI Tools: [86, 90, 108]

Signature Extraction: [47, 78, 85]

Evaluation of AV tools: [18, 26, 86]

Case Study/Synopsis: [2-5, 33, 41, 64, 66, 69, 76, 77, 105, 110, 127]

Trend: [13, 43, 55, 67, 95, 106, 107, 113]

Propagation models/Epidemiology:[20, 41, 52, 53, 69, 96, 101, 111, 118, 126, 127]

Buffer Overflow: [8, 23, 24, 31, 36, 40, 45, 59, 60, 66, 74, 81, 94, 109]

Experiment/Simulation: [98, 101, 111, 115]

Miscellaneous: [12, 17, 27, 38, 42, 65, 72, 83, 88, 89, 91, 117]

Bibliography

[1] "The Digital Immune System," Symantec, http://securityresponse.symantec.com/avcenter/reference/dis.tech.brief.pdf, Last accessed.

[2] "W32.Cabanas," http://securityresponse.symantec.com/avcenter/venc/data/w32.cabanas.html, Last accessed.

[3] "W32/Chiton," http://www.virusbtn.com/resources/viruses/indepth/gemini.xml, Last accessed.

[4] "W32/Gemini," http://www.virusbtn.com/resources/viruses/indepth/gemini.xml, Last accessed.

[5] "W95.Bistro," http://securityresponse.symantec.com/avcenter/venc/data/w95.bistro.html, Last accessed.

[6] L. M. Adleman, "An Abstract Theory of Computer Viruses," in *Advances in Computing - Crypto'88*, 1988.
This paper applies formal computability theory to viruses. It presents definition for computer viruses based on set theory. Viruses have been broken up into benign, disseminating, malicious, and Epeian categories. It proves that "detecting viruses is quite untractable". It identifies several areas of possible research including complexity theoretic and program size theoretic aspects of computer viruses, protection mechanisms and development of other models.

[7] B. Barak, et al., "On the (Im)Possibility of Obfuscating Programs," in *Advances in*

Cryptology (CRYPTO'01), Santa Barbara, California, 2001.
The paper rules out as impossible the following notion of obfuscation: An obfuscator is an efficient probabilistic program that takes as input a program P and produces as output a program O(P) such that O(P) computes the same function as P and "[...] anything one can efficiently compute from the obfuscated program [code and executable], one should be able to efficiently compute given just oracle access to the program." This means that having access to a description of O(P) should be no better than having access to P as a black box (i.e. The only efficiently understandable/ analyzable part of O(P) is the output).
That is, all the information one can get from O(P) can be as easily obtained by running the black box implementing P and taking note of the output. And no analysis of the description of the obfuscated program can efficiently yield results that cannot be efficiently obtained from the black box.

[8] A. Baratloo, N. Singh, and T. Tsai, "Transparent Run-Time Defense against Stack Smashing Attacks," in *Proceedings of the USENIX Annual Technical Conference*, 2000.

[9] J. Bergeron, et al., "Static Detection of Malicious Code in Executable Programs," in *Symposium on Requirements Engineering for Information Security (SREIS'01)*, 2001.
This paper approaches the problem of detection of malicious code in executable programs using static analysis. It involves three steps: the generation of intermediate representation, analyzing the control and data flows, and then doing static verification. Static verification consists of comparing a security policy to the output of the analysis phase. A brief description of a prototype tool is also given.

[10] J. Bergeron, et al., "Static Analysis of Binary Code to Isolate Malicious Behaviors," in *IEEE 4th International Workshop on Enterprise Security (WETICE'99)*, Stanford University, California, USA, 1999.
This paper addresses the problem of static slicing on binary executables for the purpose of detecting malicious code in commercial off-the shelf software components. The paper first defines a malicious code. To analyze malicious code, the executable is first disassembled and passed through a series of transformations. These transformations aid in getting a high level imperative representation of the code. This leads to improved analyzability while preserving the original semantics. Next, the program is sliced to extract code segments critical from standpoint of security. The behavior of these segments is reviewed for malicious characteristics.

[11] M. Bishop, "An Overview of Computer Viruses in a Research Environment," 1992.
This paper analyzes virus in a general framework. A brief history of computer viruses is presented and any presence of threat relevant to research and development systems has been investigated. It examines several specific areas on vulnerability in research-oriented systems.

[12] V. Bontchev, "Analysis and Maintenance of a Clean Virus Library," in *3 rd Int. Virus Bull. Conf*, 1993.
This provides the methods adopted to facilitate the maintenance of large amounts of different virus samples for the sake of anti-virus research. The paper presents guidelines and procedures used to maintain virus collection at the university of Hamburg's Virus Test Center.

[13] V. Bontchev, "Future Trends in Virus Writing," in *International Virus Bulletin Conference*, 1994.
This paper summarizes some ideas that are likely to be used by virus writers in the future and suggests the kind of measures that could be taken against them.

[14] V. Bontchev, "Possible Virus Attacks against Integrity Programs and How to Prevent Them," in *Proceedings of the 6th International Virus Bulletin Conference*, 1996.

This paper discusses the ways of attacking one of the most powerful methods of virus detection on integrity checking programs. It demonstrates what can be done against these attacks.

[15] V. Bontchev, "Macro Virus Identification Problems," in *7th International Virus Bulletin. Conference*, 1997.
This paper discusses some interesting theoretical problems to anti-virus software. Two viral sets of macros can have common subsets or one of the sets could be a subset of the other. The paper discusses the problems caused by this. It emphasizes the difficulties that could be exploited by the virus writers and methods, which could be followed to tackle it.

[16] V. Bontchev, *Methodology of Computer Anti-Virus Research*, Faculty of Informatics, University of Hamburg Thesis, 1998.
This thesis is a detailed writing on computer viruses. It can be treated as a definitive text on understanding and dealing with computer viruses. The important topics discussed in this work include classification and analysis of computer viruses, state of art in anti-virus software, possible attacks against anti-virus software, test methods for anti-virus software systems and social aspects of virus problem. It also discusses useful applications of self-replicating software.

[17] V. Bontchev, "The "Pros" and "Cons" of Wordbasic Virus Upconversion," in *8th International Virus Bulletin Conference*, 1998.
This paper discusses the ethical problem faced by anti-virus researchers due to the automatic Upconversion of WordBasic Viruses to Visual Basic for Applications version 5. Since a macro virus written in one language has been automatically converted to another language it is yet another unique virus. Due to this inherent feature of MS Office 97, virus researchers have to create new virus to prepare an antidote. A side effect of this activity has reportedly been that these upconverts are created and "officially" listed as existing in some anti-virus product stimulates their creation and distribution by the virus exchange people. The author has given suggested solutions for this problem.

[18] V. Bontchev, "Vircing the Invircible," http://www.claws-and-paws.com/virus/papers/, Last accessed 11/05/2004.

[19] F. Castaneda, E. C. Sezer, and J. Xuy, "Worm Vs. Worm: Preliminary Study of an Active Counter-Attack Mechanism," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
This paper proposes a method that transforms a malicious worm into an anti-worm which disinfects its original and evaluate the method using the CodeRed, Blaster and Slammer worms.

[20] D. Chess, "Future of Viruses on the Internet," in *Virus Bulletin Conference*, San Francisco, California, 1997.
This paper discusses the role of the Internet in the Virus problem. It reasons for the availability of better-equipped crisis teams that may arise due to the continued growth of the Internet. Integrated mail systems and the rise in mobile program systems on the Internet have impacted the trends in virus spread. The deployment of network aware software systems on the Internet has contributed positively to the spread of network-aware virus. The paper briefly lists some generic features of the software, which aid in virus spread.

[21] D. M. Chess, "Virus Verification and Removal Tools and Techniques." High Integrity Computing Lab, IBM T. J. Watson Research Center, Post Office Box 218, Yorktown Heights, NY, USA, 1991.
This paper describes VERV, A Prototype Virus Verifier and Remover, and a Virus Description Language for VERV.

[22] D. M. Chess and S. R. White, "An Undetectable Computer Virus," in *Virus Bulletin Conference*, 2000.

This paper extends Fred Cohen's demonstration on computer Viruses that there is no algorithm that can perfectly detect all possible viruses. This paper points out that there are computer viruses, which no algorithm can detect, even under somewhat more liberal definition of detection.

[23] E. Chien and P. Szor, "Blended Attacks - Exploits, Vulnerabilities and Buffer-Overflow Techniques in Computer Viruses," in *Virus Bulletin Conference*, New Orleans, USA, 2002.
In this paper, the authors cover such techniques as buffer overflows and input validation exploits, plus how computer viruses are using them to their advantage. The authors also discuss tools, techniques and methods to prevent these blended threats.

[24] T.-C. Chiueh and F.-H. Hsu, "Rad: A Compile-Time Solution to Buffer Overflow Attacks," in *International Conference on Distributed Computing Systems (ICDCS),*, Phoenix, Arizona, USA,, 2001.
Return Address Defender (RAD) automatically creates a safe area to store a copy of return addresses to defend programs against buffer overflow attacks. It also automatically adds protection code into applications that it compiled. Using it to protect a program does not need to modify the source code of the program. Besides, RAD does not change the layout of stack frames, so binary code it generated is compatible with existing libraries and other object files. .

[25] M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," in *12th USENIX Security Symposium*, Washington, D.C, 2003.
Techniques exist that attempt to foil the disassembly process. These techniques are very effective against state-of-the-art disassemblers, preventing a substantial fraction of a binary program from being disassembled correctly. This could allow an attacker to hide malicious code from static analysis tools that depend on correct disassembler output (such as virus scanners). The paper presents novel binary analysis techniques that substantially improve the success of the disassembly process when confronted with obfuscated binaries. Based on control flow graph information and statistical methods, a large fraction of the program's instructions can be correctly identified. An evaluation of the accuracy and the performance of our tool is provided, along with a comparison to several state-of-the-art disassemblers

[26] M. Christodorescu and S. Jha, "Testing Malware Detectors," in *Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis*, 2004.
This paper presents a technique based on program obfuscation for generating tests for malware detectors. Three widely-used commercial virus scanners have been evaluated and it is shown that the resilience of these scanners to various obfuscations is very poor.

[27] F. Cohen, "Computer Viruses-Theory and Experiments," *Computers and Security*, 6, 1984This paper brought the term "computer viruses" to general attention. It describes computer viruses and also describes several experiments in each of which all system rights were granted to an attacker in under an hour.

[28] F. Cohen, *Computer Viruses*, University of Southern California Thesis, 1985.
This is the first formal work in the field of computer viruses.

[29] F. Cohen, "Computational Aspects of Computer Viruses," *Computers and Security*, vol. 8, pp. 325, 1989.
It presents a model for defining computer viruses. It formally defines a class of sets of transitive integrity-corrupting mechanisms called "viral-sets" and explores some of their computational properties.

[30] F. Cohen, "A Formal Definition of Computer Worms and Some Related Results," *Computers and Security*, vol. 11, pp. 641-652, 1992.

A formal definition for computer worms has been presented. The definition is based on Turing's model of computation.

[31] C. Cowan, et al., "Stackguard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks," in *Proceedings of the 7th USENIX Security Conference*, San Antonio, TX, 1998.
This paper presents StackGuard, a systematic solution to the buffer overflow problem. StackGuard is a simple compiler extension that limits the amount of damage that a buffer overflow attack can inflict on a program. Programs compiled with StackGuard are safe from buffer overflow attack, regardless of the software engineering quality of the program.

[32] M. Debbabi, "Dynamic Monitoring of Malicious Activity in Software Systems," in *Symposium on Requirements Engineering for Information Security (SREIS'01)*, Indianapolis, Indiana, USA, 2001.
The authors discuss a dynamic monitoring mechanism, comprising of a watchdog system, which dynamically enforces a security policy. The authors reason this approach by stating that static analysis technique will not be able to detect malicious code inserted after the analysis has been completed. This paper discusses a dynamic monitor called DaMon. This is capable of stopping certain malicious actions based on the combined accesses to critical resources (files, communication ports, registry, processes and threads) according to rudimentary specifications.

[33] M. W. Eichin and J. A. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," in *IEEE Symposium on Research in Security and Privacy*, 1989.
In early November 1988 the Internet, a collection of networks consisting of 60,000 host computers implementing the TCP/IP protocol suite, was attacked by a virus, a program which broke into computers on the network and which spread from one machine to another. This paper is a detailed analysis of the virus program itself, a detailed routine by routine description of the virus program including the contents of its built in dictionary is provided.

[34] D. Ellis, "Worm Anatomy and Model," in *2003 ACM workshop on Rapid Malcode.*
This paper presents a general framework for reasoning about network worms and analyzing the potency of worms within a specific network. Based on a survey of contemporary worms it develops a relational model that associates worm parameters, attributes of the environment, and the subsequent potency of the worm. It then provides a worm analytic framework that captures the generalized mechanical process a worm goes through while moving through a specific environment and its state as it does so.

[35] D. Ellis, et al., "A Behavioral Approach to Worm Detection," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
This paper presents an approach to the automatic detection of worms using behavioral signatures.

[36] H. Etoh and K. Yoda, "Protecting from Stack-Smashing Attacks," IBM Research, Tokyo, http://www.trl.ibm.com/projects/security/ssp/main.html, Last accessed January 13.

[37] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer Immunology," *Communications of the ACM*, 1996.
This papers gives an overview of how the natural immune system relates to computer security and then illustrates these ideas with two examples.

[38] R. B. Fried, "A System Administrator's Guide to Implementing Various Anti-Virus Mechanisms: What to Do When a Virus Is Suspected on a Computer Network," http://www.sans.org/rr/whitepapers/malicious/43.php, Last accessed.

[39] Y. G. D. George I. Davida, and Brian J. Matt, "Defending Systems against Viruses through Cryptographic Authentication.," in *IEEE Symposium on Computer Security and Privacy*, 1989.
This paper describes the use of cryptographic authentication for controlling computer viruses. The objective is to protect against viruses infecting software distributions, updates, and programs stored or executed on a system. The authentication scheme determines the source and integrity of an executable, relying on the source to produce virus-free software. The scheme presented relies on a trusted device, the authenticator, used to authenticate and update programs and convert programs between the various formats. In addition, each user's machine uses a similar device to perform run-time checking.

[40] A. K. Ghosh and T. O'Connor, "Analyzing Programs for Vulnerability to Buffer Overrun Attacks," in *Proc. 21st NIST-NCSC National Information Systems Security Conference*, 1998.
Determines whether a program has buffer overflow by dynamic analysis. Uses a fault injection method for this purpose. An analyst manually searches for buffers. Then introduces buffer overflow function. If there is a buffer overflow this code is executed.

[41] D. Hanson, et al., "A Comparison Study of Three Worm Families and Their Propagation in a Network," http://www.securityfocus.com/infocus/1752, Last accessed.

[42] V. Heavens, "Virus Creation Tools," http://vx.netlux.org/dat/vct.shtml, Last accessed 08/29/2003.

[43] J. D. Howard, *An Analysis of Security Incidents on the Internet 1989-1995*, Carnegie Institute of Technology, Ph.D Dissertation Thesis, 1997.
This dissertation analyses the trends in the Internet Security by investigating 4,299 security-related incidents on the Internet reported to the CERT Coordination Center (CERT/CC) from 1989 to 1995.

[44] J. Hruska, "Computer Virus Prevention: A Primer," Sophos Labs, http://www.sophos.com/virusinfo/whitepapers/prevention.html, Last accessed 08/29/2003.

[45] F.-H. Hsu, "The Principle, Attack Patterns, and Defense Methods of Buffer Overflow Attacks," State University of New York at Stony Brook, Stony Brook, RPE TR-87, October 2000 2000.
This paper presents a solution to the buffer overflow attack problem using which users can prevent attackers from compromising their systems by changing the return address to execute injected code, which is the most common method used in buffer overflow attacks.

[46] M. D. e. a. J. Bergeron, "Detection of Malicious Code in Cots Software: A Short Survey," in *First International Software Assurance Certification Conference (ISACC'99)*, Washington DC, 1999.
This paper describes the main characteristics of malicious code and proposes taxonomy for the existing varieties. A formal definition of malicious code has been given. A new taxonomy that is oriented towards the goal of detecting malicious code has been defined. Different static, dynamic analysis methods and ad hoc techniques have been discussed. It discusses several techniques to detect malicious code in commercial-off-the-shelf software products. The paper concludes by looking at the advantages and disadvantages of static analysis over dynamic analysis methods.

[47] B. A. Jeffrey O. Kephart, "Automatic Extraction of Computer Virus Signatures," in *4th Virus Bulletin International Conference*, 1994.
This paper discusses the idea of automatically identifying viral signatures from machine code using statistical methods.

[48] G. B. S. Jeffrey O. Kephart, Morton Swimmer, and Steve R. White, "Blueprint for a Computer Immune System," in *Virus Bulletin International Conference*, 1997.

Since the internet will provide a fertile medium for new breeds of computer viruses, the authors have described a immune system for computers that senses the presence of a previously unknown pathogen that within minutes, automatically derives and deploys a prescription for detecting and removing the pathogen.

[49] M. Jordan, "Anti-Virus Research-Dealing with Metamorphism," *Virus Bulletin*, 2002The paper discusses some observation of the properties of metamorphic viruses and provides a possible method that AV scanners can use to deal with metamorphism.

[50] M. Jordon, "Dealing with Metamorphism," *Virus Bulletin*, 2002This article will discuss one possible method that AV scanners could use to deal with metamorphism.

[51] J. O. Kephart and B. Arnold, "A Biologically Inspired Immune System for Computers," in *Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994.
An immune system for computers and computer networks is designed that takes much of its inspiration from nature. Like the vertebrate immune system this system develops antibodies to previously unencountered computer viruses or worms and remembers them so as to recognize and respond to them more quickly in the future.

[52] J. O. Kephart and S. R. White, "Directed-Graph Epidemiological Models of Computer Viruses," in *IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, California, 1991.
This paper presents a detailed study of computer virus epidemics. It presents a theoretical view of the viral propagation using deterministic and stochastic approaches. It studies the conditions under which viral epidemics are likely to occur. It argues that an imperfect defense against a computer virus can still be highly effective in preventing widespread propagation provided that infection rate does not exceed a well-defined threshold.

[53] J. O. Kephart and S. R. White, "Measuring and Modeling Computer Virus Prevalence," in *Proceedings of the 1999 IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, California, 1993.
This paper introduces two new epidemiological models of computer virus spread. Only a small fraction of all well-known viruses have appeared in real incidents, partly because many viruses are below the theoretical epidemic threshold. Models of localized software exchange can explain the observed sub-exponential rate of viral spread.

[54] P. Kerchen, et al., "Static Analysis Virus Detection Tools for Unix Systems," in *13th National Computer Security Conference*, 1990.
This paper proposes two heuristic tools the use static analysis and verification techniques for detecting computer viruses in a UNIX environment. The tools should be used to detect infected programs before their installation. The first tool, "detector", searches for duplicate system calls in the compiled and linked program, the second tool, "Filter", uses static analysis to determine all of the files, which a program may write to. By finding out the files to which the program can or cannot write, the program can be identified as a malicious or benign.

[55] D. M. Kienzle and M. C. Elder, "Recent Worms: A Survey and Trends," in *Proceedings of the 2003 ACM workshop on Rapid Malcode*, 2003.
This paper presents a broad overview and trend of recent worm activity.

[56] C. Ko, G. Fink, and K. Levitt, "Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring," in *10 th Annual Computer Security Application Conf*, Orlando, FL, 1994.
This paper uses concepts of solving Intrusion Detection Problems to detect vulnerabilities in

programs during execution. Since the intended behaviors of privileged programs are benign, a program policy has been developed to describe this behavior, using a program policy specification language. Specifications of privileged programs in Unix have been presented, along with a prototype execution monitor, to analyze the audit trails with respect to this specification.

[57] A. Lakhotia and E. U. Kumar, "Abstract Stack Graph to Detect Obfuscated Calls in Binaries," in *Fourth IEEE International Workshop on Source Code Analysis and Manipulation(SCAM'04)*, Chicago, Illinois, 2004.
This paper presents a method to statically detect obfuscated calls in binary code. The notion of abstract stack is introduced to associate each element in the stack to the instruction that pushes the element. An abstract stack graph is a concise representation of all abstract stacks at every point in the program. An abstract stack graph, created by abstract interpretation of the binary executables, may be used to detect obfuscated calls and other stack related obfuscations.

[58] A. Lakhotia and P. K. Singh, "Challenges in Getting 'Formal' with Viruses," *Virus Bulletin*, 2003Here a staged architecture for binary malware analysis is proposed. It discusses issues relating to formal analysis methods used by optimizing compilers and other programming tools, being applied in the detection of metamorphic viruses, are not directly suitable for use in anti-virus technologies. Unlike in the context of optimizing compilers and other similar tools, the analysis tool and the programmer (virus writer) do not have a common objective. Hence, assumptions made by analysis methods used by such compilers can be exploited by the virus writer. The authors discuss how a virus writer could attack the various stages in the decompilation of binaries by taking advantage of the limitation of static analysis. Concerning metamorphic viruses the authors observe that though these viruses pose a serious challenge to anti-virus technologies, the virus writers are confronted with the same theoretical limitations and have to address some of the same challenges that the anti-virus technologies face.

[59] D. Larochelle and D. Evans, "Statically Detecting Likely Buffer Overflow Vulnerabilities," in *Proceedings of the 2001 USENIX Security Symposium*, Washington, D.C., 2001.
This paper presents a new approach to mitigating buffer overflow vulnerabilities by detecting likely vulnerabilities through an analysis of the program source code. The approach exploits information provided in semantic comments and uses lightweight and efficient static analyses.

[60] G. Lee and A. Tyagi, "Encoded Program Counter: Self-Protection from Buffer Overflow Attacks," in *Proceedings of the International Conference on Internet Computing (IC'2000)*, Las Vegas, Nevada, USA, 2000.

[61] C. Linn, S. Debray, and J. Kececioglu, "Enhancing Software Tamper-Resistance Via Stealthy Address Computations," in *19th Annual Computer Security Applications Conference (ACSAC)*, Las Vegas, Nevada, 2003.
This paper proposes new method to make it harder for an attacker to steal intellectual property and to breach the software security in order to find vulnerabilities. In order to make software 'tamper-proof' verification tools use check-sum code technique to check the validity of software. However, attackers can easily identify the check-sum code and disable it. The proposed method makes the binary harder for the attackers to identify the check-sum code and disable them. The idea is of indirection in which direct control transfers such as call and jump instructions are replaced with calls to specialized functions that are responsible for directing control to the intended targets in some stealthy manner. The intent is that the successors of each transformed basic block will be difficult to discover. The specialized functions are termed as branch functions.

[62] C. Linn, et al., "A Multi Faceted Defense Mechanism against Code Injection Attacks," http://www.cs.arizona.edu/people/debray/papers/injection-attacks.html, Last accessed.

[63] R. W. Lo, K. N. Levitt, and R. A. Olsson, "Mcf: A Malicious Code Filter," *Computers & Security*, vol. 14, pp. 541-566, 1995.
This paper discusses a programmable static Analysis tool called "Malicious Code Filter, MCF, to detect malicious code and security related vulnerabilities in system programs. The MCF uses telltale signs to determine whether a program is malicious without requiring a programmer to provide a formal specification. Program slicing techniques are used to reason about telltale malicious properties. By combining the telltale sign approach with program slicing, a small subset of a large program can be examined for malicious behavior. The paper also discusses how the approach can be defeated and then discusses a countermeasure.

[64] A. Marinescu, " An Analysis of Simile," http://www.securityfocus.com/infocus/1671, Last accessed.

[65] J. Martin, "A Practical Guide to Enterprise Antivirus and Malware Protection," http://www.sans.org/rr/whitepapers/malicious/68.php, Last accessed.

[66] B. McCorkendale and P. Szor, "Code Red Buffer Overflow," *Virus Bulletin*, 2001, http://www.peterszor.com/codered.pdf.
Having encountered conflicting information from a variety of sources about the Code Red (aka W32/Bady.worm) buffer overflow technique, Bruce McCorkendale and Péter Ször decided to look into the buffer overflow to uncover the details

[67] T. Micro, "Activex and Java: The Next Virus Carriers?," http://vx.netlux.org/lib/static/vdat/epactive.htm, Last accessed.

[68] M. Mohammed, *Zeroing in on Metamorpic Computer Viruses*, Center for Advanced Computer Studies, University of Louisiana at Lafayette, M.S. Thesis, 2003.
The thesis describes a technique to undo certain obfuscation transformations, such as statement reordering, variable renaming, and expression reshaping which are applied to hide information, in the case of malicious code. The technique applies certain counter transformation, called zeroing transformation, to detect metamorphic viruses that obfuscate their code using these techniques. Zeroing transformations zero the effect of these transformations to transform all the metamorphic variants of the virus to a single form called the zero form of the virus. The zero form of a program is based on the string representations for the statements in the program.

[69] D. Moore, C. Shannon, and K. Claffy, "Code-Red: A Case Study on the Spread and Victims of an Internet Worm," in *The second ACM SIGCOMM Workshop on Internet measurment*.
This paper details the spread of the Code-Red and CodeRedII worms in terms of infection and deactivation rates and examines the properties of the infected host population, including geographic location, weekly and diurnal time effects, top-level domains, and ISPs

[70] J. F. Morar and D. M. Chess, "Can Cryptography Prevent Computer Viruses?," in *Virus Bulletin Conference*, 2000.
The relationship between cryptography and virus prevention is complex. Solutions to the virus prevention problem involving cryptography have been proposed, though these solutions do not contribute much to the prevention techniques prevalent at present. This paper discusses the role of encryption in the field of virus authoring and in the field of Anti-Virus research.

[71] J. Munro, "Antivirus Research and Detection Techniques," http://www.extremetech.com/article2/0,1558,325439,00.asp?rsDis=Antivirus_Research_and_Detection_Techniques,_Part_I-Page001-28716, Last accessed.

[72] C. Nachenberg, "Computer Virus-Antivirus Coevolution," *Communications of the ACM*, vol. 40, 1997.
This is an overview of the current status of the conflict between virus writers and AV community. As antivirus programs became more effective, virus writers countered by developing techniques that allowed their products to escape detection. "coevolution" is the author's apt term for the never-ending threat-response cycle in which attackers and defenders engage. The author ends with some speculation about the inevitable next rounds in this war without end.

[73] L. Oudot, "Fighting Internet Worms with Honeypots," 2003.

[74] P. Pathak, *Linux Reloaded: A Linux System Hardened against Buffer Overflow Attacks*, The Center for Advanced Computer Studies, University of Louisiana at Lafayette, M.S. Thesis Thesis, 2003.
A comprehensive classification of buffer overflow attacks is done and a novel approach for the detection and prevention of various types of buffer overflow attacks for segmented architectures has been developed. The approach prevents buffer overflow attacks at the kernel and compiler level. O.S based and compiler based approaches are integrated in order to gain benefits of both. "Linux Reloaded" (LR), an enhanced Linux System is developed by modifying the Linux kernel and the GCC compiler

[75] S. Pearce, "Viral Polymorphism," http://vx.netlux.org/lib/asp00.html, Last accessed.

[76] R. Pethia, "The Melissa Virus: Inoculating Our Information Technology from Emerging Threats," CERT, Carnegie Mellon University 1999.
Provides a detail discussion on Melissa.

[77] J. Phillips, "Overview of Nimda," http://www.sans.org/rr/whitepapers/malicious/92.php, Last accessed.

[78] S. Popinsky, "Advanced Clamav Signatures," http://www.antionline.com/showthread.php?threadid=262564&pagenumber=1, Last accessed Nov. 4, 2004.

[79] P. Porras, et al., "A Hybrid Quarantine Defense," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
This paper studies the strengths, weaknesses, and potential synergies of two complementary worm quarantine defense strategies under various worm attack profiles.

[80] M. M. Pozzo and T. E. Gray, "An Approach to Containing Computer Viruses," *Computers and Security*, vol. 6, pp. 321-331, 1987.
This paper presents a mechanism for containing the spread of computer viruses by detecting at run-time whether or not an executable has been modified since its installation. The detection strategy uses encryption and is held to be better for virus containment than conventional computer security mechanisms, which are based on the incorrect assumption that preventing modification of executables by unauthorized users is sufficient. Although this detection mechanism is most effective when all the executable on a system are encrypted, a scheme is presented that shows the usefulness of the encryption approach when this is not the case.

[81] M. Prasad and T.-C. Chiueh, "A Binary Rewriting Defense against Stack Based Buffer Overflow Attacks," in *USENIX 2003 Annual Technical Conference*, 2003.
Buffer overflow attack is the most common and arguably the most dangerous attack method used in Internet security breach incidents reported in the public literature. The work reported in this paper explores application of static binary translation to protect Internet software from buffer overflow attacks.

[82] J. C. Rabek, et al., "Defensive Technology: Detection of Injected,

Dynamically Generated, and Obfuscated Malicious Code," in *ACM workshop on Rapid Malcode*, 2003.
This paper presents DOME, a host-based technique for detecting several general classes of malicious code in software executables. DOME uses static analysis to identify the locations (virtual addresses) of system calls within the software executables, and then monitors the executables at runtime to verify that every observed system call is made from a location identified using static analysis.

[83] C. Raiu, " A Virus by Any Other Name: Virus Naming Practices," http://www.securityfocus.com/infocus/1587, Last accessed.

[84] J. Reynolds, "Rfc1135: The Helminthiasis of the Internet," http://ftp.arnes.si/standards/rfc/rfc1135.txt, Last accessed.

[85] E. S. Sandeep Kumar, "Generic Virus Scanner in C++," in *8 th Computer Security Applications Conference*.
This paper discusses a generic virus detection tool designed for recognizing viruses across different platforms. The paper initially discusses various methods of virus detection and then describes a generic signature scanner as an anti-virus tool.

[86] A. Saxena and A. Dwivedi, "A Novel Approach to Improve Scanners Using Fuzzy Logic," in *Proceedings of 2004 Virus Bulletin Conference*, Chicago, 2004.
Conventional method of testing AV scanners does not take into account the characteristics of the virus. They focus on how many viruses are missed without taking into account the damage caused by the virus. The paper proposes a Fuzzified Scanner Comparison Technique ranks the viruses on three variables: time to spread, the severity of damage, and operating system(s) attacked. A ranked list of viruses is then created for testing and evaluating AV scanners.

[87] S. E. Schechter, J. Jung, and A. W. Berger, "Fast Detection of Scanning Worm Infections," in *7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, French Riviera, France, 2004.
This presents a hybrid approach to detecting scanning worms.

[88] S. E. Schechter and M. D. Smith, "Access for Sale: A New Class of Worm," in *2003 ACM workshop on Rapid Malcode*.
This paper introduces a new type of worm that enables division of labor, installing a back door on each infected system that opens only when presented a system-specific ticket generated by the worm's author. In addition to describing this new threat, it proposes a number of approaches for defending against it.

[89] F. B. Schneider, "Enforceable Security Policies," *Information and System Security*, vol. Vol. 3 (No. 1), pp. pp. 30--50, 2000.
A precise characterization is given for the class of security policies enforceable with mechanisms that work by monitoring system execution. Security automata are introduced for specifying exactly the class of security policies discussed. Techniques to enforce security policies specified by such automata are also discussed.

[90] M. G. Schultz, et al., "Data Mining Methods for Detection of New Malicious Executables," in *IEEE Symposium on* Security and Privacy, 2001.
This paper presents a framework for detection of malicious executables with viral characteristics. Since signatures for new viruses are not known, the data mining technique presented in this work will be able to solve this problem in a better way than the current signature-based methods of virus detection.

[91] J. F. Shoch and J. A. Hupp, "The "Worm" Programs- Early Experience with a Distributed Computation," *CACM*, vol. 25, pp. 172-180, 1982.
This is an exploratory paper for its time. This paper discusses issues found in the early

exploration of distributed computing. Authors talk about the motivations and definitions for a worm program from the distributed computation perspective. Not much work had been done in building distributed systems in 1982.

[92] P. K. Singh, *A Physiological Decomposition of Virus and Worm Programs*, CACS, University of Louisiana, Lafayette, Master Thesis Thesis, 2002.
Components of malicious code formally defined.

[93] P. K. Singh, M. Moinuddin, and A. Lakhotia, "Using Static Analysis and Verification for Analyzing Virus and Worm Programs," in *Proceedings of the 2nd European Conference on Information Warfare and Security*, Reading, UK, 2003.
A framework for statically verifying binary executables for malicious code has been described. A decomposition of virus and worm programs based on their functional components has been provided.

[94] N. Smith, "Stack Smashing Vulnerabilities in the Unix Operating System," Southern Connecticut State University, http://destroy.net/machines/security/nate-buffer.ps, Last accessed.

[95] A. Solomon, "A Brief History of Pc Viruses," Last accessed.

[96] A. Solomon, "Epidemiology and Computer Viruses," http://vx.netlux.org/lib/static/vdat/epepidem.htm, Last accessed.

[97] E. H. Spafford, "The Internet Worm Program: An Analysis," *Computer communication review*, vol. 19, pp. 17-57, 1989.
This paper is an analytical commentary on the Internet Worm program, which infected the Internet on the evening of November 2nd 1988. The paper defines Worms and Viruses. It discusses the flaws in computer systems that were exploited by the Worm to spread across the Internet. Patches to these flaws are also discussed. A high level description of the functioning of the Worm program is also provided. The paper then carries a detailed analysis of the Worm.

[98] E. H. Spafford, "Computer Viruses as Artificial Life," *Artificial Life*, vol. 1, pp. 249-265, 1994.
This paper talks about how computer viruses operate, their history, and the various ways computer viruses are structured. It then examines how viruses meet properties associated with life as defined by some researchers in the area of artificial life and self-organizing systems. The paper concludes with some comments directed towards the definition of artificially "alive" systems and related experiments.

[99] D. J. Stang, "Fighting Computer Virus Infection through Auto-Immune Responses." http://vx.netlux.org/lib/static/vdat/epautoim.htm

[100] D. J. Stang, "Virus Prevention Policy," http://vx.netlux.org/lib/static/vdat/virpolic.htm, Last accessed.

[101] S. Staniford, et al., "The Top Speed of Flash Worms," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
Simulating a flash version of Slammer, calibrated by current Internet latency measurements and observed worm packet delivery rates, this paper shows that a worm could saturate 95% of one million vulnerable hosts on the Internet in 510 milliseconds. A similar worm using a TCP based service could 95% saturate in 1.3 seconds.

[102] G. Szappanos, "Are There Any Polymorphic Macro Viruses at All? (… and What to Do with Them)," in *Proceedings of the 12th International Virus Bulletin Conference*, 2002.

It discusses how polymorphic macro viruses fit into binary polymorphic viruses and about their detection.

[103] G. Szappanos, "Polymorphic Macro Viruses, Part One," Security Focus, http://online.securityfocus.com/infocus/1635, Last accessed 08/29/2003.

[104] P. Szor, "Generic Disinfection," http://vx.netlux.org/lib/static/vdat/epgendis.htm, Last accessed.

[105] P. Szor, "Coping with Cabanas," *Virus Bulletin*, pp. 10-12, 1997.
This is an analysis of cabanas.

[106] P. Ször, "Attacks on Win32 - Part Ii," in *Virus Bulletin Conference September 2000*, Orlando, FL, 2000.
In 1998 several anti-virus companies introduced heuristic scanning for 32-bit Windows viruses. As a result the number of anti-heuristic viruses is on the rise. This paper introduces infection methods with special attention to the anti-heuristic infection techniques. It also provides results achieved by testing old Win32 viruses and worms on Windows 2000. It also says about possible new virus models likely to be seen in the near future.

[107] P. Ször and P. Ferrie, "Hunting for Metamorphic," in *Virus Bulletin Conference*, Prague, Czech Republic, 2001.
The paper shows the trend of evolution of complex viruses such as polymorphic and metamorphic. Detailed survey of various metamorphic engines and the techniques used by them is presented. The article aims to provide a better understanding of problems faced by AV community. Some examples are also given that shows techniques of detecting new generation metamorphic viruses.

[108] G. Tesauro, J. O. Kephart, and G. B. Sorkin, "Neural Network for Computer Virus Recognition," *IEEE Expert*, vol. 11, pp. 5-6, 1996.
This paper describes a neural network for generic detection of boot sector viruses that infect the boot sector of a floppy disk or a hard drive.

[109] T. K. Tsai and N. Singh, "Libsafe: Protecting Critical Elements of Stacks," Avaya Labs Research, ALR-2001-019, 2001.
This paper presents a method to detect and handle exploitation of buffer overflow vulnerabilities. This method works with any existing pre-compiled executable and can be used transparently, even on a system-wide basis.

[110] M. Venable, P. Prathak, and A. Lakhotia, "Getting inside Beagle's Backdoor," *Virus Bulletin*, 2004This article presents a detailed analysis of a backdoor contained within the W32/Beagle.J worm. The focus is on the capabilities and protocol of the backdoor, as well as the process used to uncover these capabilities

[111] A. Wagner, et al., "Experiences with Worm Propagation Simulations," in *ACM workshop on Rapid Malcode*, 2003.
This paper describes the design of a simulator and compares observation of past worms with simulated behaviour. One specific feature of the simulator is that the Internet model used can represent network bandwidth and latency constraints.

[112] C. Wang, J. C. Knight, and M. C. Elder, "On Computer Viral Infection and the Effect of Immunization," ftp.cs.virginia.edu/pub/techreports/CS-99-32.pdf, Last accessed.

[113] N. Weaver, "A Brief History of the Worm," http://www.securityfocus.com/infocus/1515, Last accessed.

[114] N. Weaver, "Future Defenses: Technologies to Stop the Unknown Attack," http://tennis.ecs.umass.edu/~czou/research/emailDefense-TR.pdf, Last accessed.

[115] N. Weaver, et al., "Preliminary Results Using Scaledown to Explore Worm Dynamics," in *Washington, DC, USA*, 2004.

This paper presents initial results from investigating scaledown techniques for approximating global Internet worm dynamics by shrinking the effective size of the network under study.

[116] N. Weaver, et al., "A Taxonomy of Computer Worms," in *ACM workshop on Rapid Malcode*, 2003.
To understand the threat posed by computer worms, it is necessary to understand the classes of worms, the attackers who may employ them, and the potential payloads. This paper describes a preliminary taxonomy based on worm target discovery and selection strategies, worm carrier mechanisms, worm activation, possible payloads, and plausible attackers who would employ a worm.

[117] I. Whalley, et al., "An Environment for Controlled Worm Replication and Analysis," http://www.research.ibm.com/antivirus/SciPapers/VB2000INW.htm, Last accessed.

[118] I. Whalley, et al., "An Environment for Controlled Worm Replication and Analysis Or: Internet-Inna-Box," in *Virus Bulletin Conference*, 2000.
The paper outlines a functional prototype of a worm replication system. Techniques and mechanisms for constructing and utilizing an environment enabling the automatic examination of worms and network bases viruses have been described. The paper involves a very brief description of some well-known worms from the past and the present. It elaborates on techniques used by worms to spread across networks. Finally an anatomy of the worm replicator system is presented.

[119] S. R. White, "Open Problems in Computer Virus Research," in *Virus Bulletin Conference*, 1998.
This paper identifies some challenging open issues on computer virus detection and protection. It lists out five problems in this field, namely, Development of New Heuristics for virus detection, the study of viral spread and epidemiology, deploying distributed digital immune system for detecting new viruses, detection of worm programs and proactive approaches towards detection of virus programs.

[120] C. Wong, et al., "A Study of Mass-Mailing Worms.," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
This paper presents an in-depth study on the effects of two mass-mailing worms, SoBig and MyDoom, on outgoing traffic and develops insight into the possibilities and challenges of automatically detecting, suppressing and stopping mass-mailing worm propagation in an enterprise network environment.

[121] J. Xiong, "Act: Attachment Chain Tracing Scheme for Email Virus Detection and Control," in *ACM Workshop on Rapid Malcode (WORM 2004)*, George Mason University, Fairfax, Virginia, USA, 2004.
This paper proposes an automated email virus detection and control scheme using attachment chain tracing (ACT) technique.

[122] T. Yetiser, "Polymorphic Viruses, Implementation, Detection and Protection," http://vx.netlux.org/lib/static/vdat/pviripd.htm , Last accessed.

[123] L. Zeltser, "Reverse Engineering Malware," http://www.zeltser.com/sans/gcih-practical/revmalw.html, Last accessed.

[124] Zombie, "Disassemblers within Viruses," http://z0mbie.host.sk/, Last accessed.

[125] Zombie, "Some Ideas About Metamorphism," http://vx.netlux.org/lib/vzo20.html, Last accessed.

[126] C. C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense," in *2003 ACM workshop on Rapid Malcode*.
Enlightened by the methods used in epidemic disease control in the real world, this paper

presents a dynamic quarantine method based on the principle "assume guilty before proven innocent" and shows that the dynamic quarantine can reduce a worm's propagation speed.

[127] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," in *9th ACM conference on Computer and communications security*, 2002.
Based on the classical epidemic Kermack-Mckendrick model, this paper derives a general Internet worm model to have a better understanding and prediction of the scale and speed of Internet worm spreading.

[128] C. C. Zou, W. Gong, and D. Towsley, "Feedback Email Worm Defense System for Enterprise Networks," University of Massachusetts, Amherst 2004.
This presents an architecture and system design of a "feedback email worm defense system" to protect email users in enterprise networks.