Abstract

Program Vulnerabilities may be unwarranted for any organization and may lead to severe system failure. Due to the advancement of technology there has been increase in the area of vulnerability attacks which are exploited by hackers for getting access to the system or insertion of their malicious code. In this paper we present a proposal for compiler design which prevents some common vulnerability. The output result for our compiler would be compile time warning stating the possible vulnerability in the code. We will also look into the details about the different type of vulnerability and how the attacker can exploit those vulnerabilities in order to corrupt the system. The knowledge of various vulnerability creation areas have been used to design a compiler for vulnerability prevention. Compiler in this publication uses the symbol table generation mechanism for syntactically, semantically segregation of executable code and canary guard mechanism for the protection of cases of buffer overflow. Major work in this area deals with the simple scenarios for vulnerability detection but our aim is to check for various complicated scenarios and non common possibilities for program attack and designing a framework preventing such kinds of attacks.
References

- Alexander Ivanov Sotirov, automatic vulnerability detection using static source code analysis.
- David Brumley, Tzi-cker Chiueh, Robert Johnson, RICH: Automatically Protecting Against Integer-Based Vulnerabilities.
- Steven Muchnick, Advance compiler design and implementation.
- James C Foster, Vitlay Osipov, Nish Bhalla, Niels Heinen, Book on Buffer overflow attack.
- K. V. N Sunitha, Book on Compiler Construction
- Alfred V. Aho, Monica S. Lam, Ravi Sethi and D. Jeffrey Ullman, Book on Compilers Principles Techniques And Tools
- M. Gschwind and E. R. Altman. Precise exception semantics in dynamic compilation.

Index Terms

Computer Science

Security

Keywords

Program vulnerabilities Stack smashing Buffer overflow Canary guard Compiler

Canaries

Terminator.