Example of essay on software security assessment

Business, Management

\n[toc title="Table of Contents"]\n

\n \t

\n[/toc]\n \n

PROGRAM BUILDING BLOCKS

Programming aspects exist in two forms: data and instructions. Working with data requires the knowledge of variables and types while working with instructions require the understanding of control structures and subroutines. These are program building blocks as they form the fundamental aspects of programming.

Variables represent the memory locations or unit assigned a name to be easily referred to and accessed by the program. The programmer only needs to remember the name and the compiler keep track of the memory location. A programmer needs to know the name referring to a certain unit or location of the memory used to store data. Different programming languages have different variables used to specify the data to hold. One variable holds

integers while others hold characters.

A program is a sequence of instructions executed by a computer in a step by step manner. In order to manage memory and instructions, computers require control structures. Control structures are distinct instructions that manage the flow of control. Control structures are classified into two types; loops and branches. Loops allow sequence instructions to be repeated over and over while branches query the computer on the course to be taken by testing the underlying conditions that occur as the program runs.

Large programs that execute complex activities might be impossible to build. However subroutines provide a way of executing complex operations. Subroutines provide instructions for performing a certain task grouped together as a unit and named. The name is then substituted to the whole set of instructions.

AUDITING VARIABLE USE

Variables are fundamental objects used to store data elements in an application. Their meaning is derived from the way they are used, what's stored in them and what operations they perform. Code auditing is the process of understanding variables and their relations in respect to each other. It also investigates how applications can be adversely affected by their unexpected manipulation of their relationships. One aspect of auditing variable use is variable relationships. Variables are related to each other if their values depend on some fashion or state. For instance two variable of which one points to a writable location in the buffer while another keeps track of the amount of space in that buffer. The two variables are related,

and their values should change in a lockstep as the buffer is manipulated. A code auditor searches for variables that are related to each other, as well as their intended relationship. It also points to a way to desynchronize these variables through a block of code that alters one variable in a way inconsistent with the other.

AUDITING CONTROL FLOW

Control flow is the process in which a processor carries out a certain sequence of instructions. Conditions, repeat instructions and subroutines enable programming languages branch to different paths. In auditing these constructs, auditors need to decipher security vulnerabilities that render programs into contexts that are not accounted for correctly. Loops and switch statement constructs govern internal control flow while functions govern external control flow.

AUDITING FUNCTIONS

Functions form a ubiquities component of modern programs irrespective of the programming language. Application programmers divide applications into functions to encapsulate functionality that can be utilized in other places of the program. It also organizes the program into small pieces that can be easily conceptualized and managed. Object oriented programmers have devised a way of creating member functions organized around objects. Auditing functions involve cognizant knowledge of the call implications such as what program changes can take place because of that change. The four main types of vulnerability patterns include return value misinterpretation,

incorrectly formatted arguments, unexpected updating of arguments, and unexpected change of a global program.

AUDITING MEMORY

Memory management is an influential element of every program whether it is executed explicitly or implicitly by the language used and runtime. A complete knowledge of the memory blocks reveals common issues in memory management and security related impacts of mismanagement.

ACC Logs

Errors in memory management are as a result of incorrect length miscalculations. Some calculations are easy to comprehend, but other are easy to miss. Allocation-Check-Copy is a tool that helps identify even the most subtle miscalculated lengths. It records the variations in allocation sizes, length checks and data element copies occurring on a memory block. ACC block has three columns. The first column describes the size of memory allocation. The second length checks the data elements are facing before being copied to the allocated buffer. The last checks the data elements copied into the buffer and the way they are copied.

ALLOCATION FUNCTIONS

A memory block is allocated with a function, and these functions are allocate memory space in bytes to the variables of different data types. The functions reserves bytes of different sizes and returns base addresses to pointer variables. The return values for memory allocation routines indicate success

or failure of the allocation process. Thus, code auditors must check the final status of memory management routines to handle errors appropriately.

ALLOCATOR SCORECARDS AND ERROR

Application review requires the identification of allocation routines early enough and identification of cursory examinations on them. Each potential danger is addressed by scoring information routines based on associated vulnerability problems. A scoreboard is used to deal with unnecessary audit logs. Error domains are used with functions and allocators to inspect a memory corruption incidence.

Reference

Bace, R. (2009). Vulnerability assessment: Computer Security Handbook . John Wiley & Sons.

Dowd M., M. J. (2006). Auditing Memory Management. Springer .

Dowd, M., McDonald, J., & Schuh, J. (2006). The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Addison-Wesley Professional.