Self Debugging Mode for Patch-Independent Nullification of Unknown Remote Process Infection

Ruo Ando and Yoshiyasu Takefuji

Keio University, Graduate School of Media and Governance,
Endo 5322, Fujisawa, 2528520 Japan
{Ruo, Takefuji}@sfc.keio.ac.jp
http://www.neuro.sfc.keio.ac.jp

Abstract. The rapid increase of software vulnerabilities shows us the limitation of patch-dependent countermeasures for malicious code. We propose a patch-independent protection technique of remote infection which enables each process to identify itself with "being infected" and nullify itself spontaneously. Our system is operating system independent and therefore does not need software rebuilding. Previously, no method for stopping malicious process without recompiling source code or rebuilding software has been proposed. In proposal system, target process is running under self debugging mode which is activated by enhancing debug() exception handler and utilizing MSR debug register. In this paper we show the effectiveness of proposal method by protecting the remote process infection without patching security holes. Implemention of device driver call back function and BranchIP recorder provides the real-time prevention of unregistered worm attack through Internet. In experiment, function test of stack buffer overflow of Win32.SQLExp.Worm is presented. Also CPU utilization corresponding to the number of calling function and some database operations is showed.

Keywords: self-debugging mode, real-time nullification, debug register, improved debug exception handler, branchIP recorder.

1 Introduction

The rapid increase of software vulnerabilities and its exploitation imposes a great burden on network administrators and client users. Recent cyber attacks, worms and virsuses become more sophisticated. Some obfuscation and avoidance techniques which evade network traffic inspection such is called polymorphic and metamorphic coding. These techniques is now applied in malicious code writing.Win32.Evol and Simile virus is valid example which shows us the limitation of naive signature based inspection of network traffic. The rapid spread of Win32.SQLExp.Worm infection also shows the limitation of patch-based coutermeasures for new attack. Previously, instead of signature matching, the adaptive prevention technique has been proposed. Some technique inspect the target file

Y.G. Desmedt et al. (Eds.): CANS 2005, LNCS 3810, pp. 85–95, 2005.

by executing it both on run-time environment and on virtual machine emulation. Another run-time protection is mainly represented by compiler solutions and operating system based method. Openwall Linux kernel patch project is to improve the protection against buffer overflows. OpenBSD also provides the new feature of stack protection technology against buffer overflows which is embedded in the system compiler. Stack-smash protection compilers are used against buffer overflows including the GCC extensions, libsafe, propolice, stackguard, libmib, and MS .net compiler. However, their disadvantage lies in that kernels and software components must be rebuilt.

In this paper we propose the method for the real-time infected process nullfication using improved debug exception handler. This could be described as automated debugging based on improved loader, driver-supplied callback function and debug exception handler. On this system attribute of self-debugging is added to the target process, which makes it possible to control by itself when attacked and infected without the file scanner or IDS.

Figure1 illustrates the concept of automated debug using improved debug exception handler. In conventional debug method, the system needs to launch debugging process and API. The target process can be executed in the memory of debugging process. In proposal method, process does not need their debugging process and API because exception handler is improved so that the debug specified for malicious code is automated in running each process.

In this paper, the proposal system is constructed on 80x86 processor. The 80x86 processor has 20 different exception handlers. Table 1 shows the exception handlers mainly concerned with debugging issues. In this paper we present improved exception handler. Particularly, we enhance debug() function with sig-

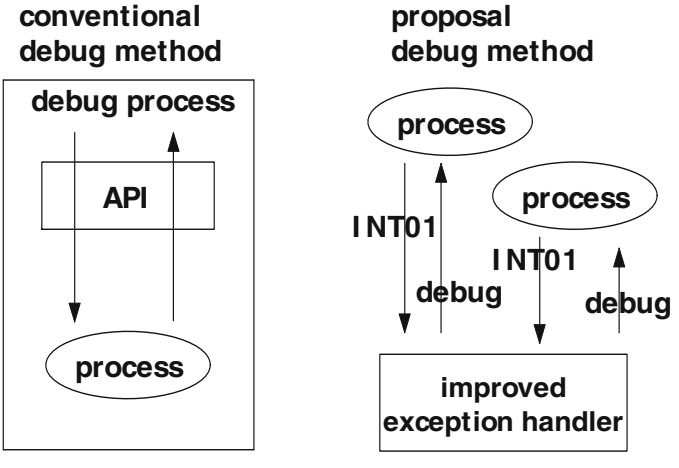

Fig. 1. Automated debugging. By improving exception handler INT01H, the self-debugging attribute is added to the target process. In proposal method, process under inspection does not need debugging process and its memory, which provides sense of infection and self-defense.