Alphanumeric Shellcode Generator for ARM Architecture

Pratik Kumar, Nagendra Chowdary, and Anish Mathuria

DA-IICT, Gandhinagar
{posani_nagendra,anish_mathuria}@daiict.ac.in

Abstract. Shellcode usually refers to a piece of code that is injected into a program in order to perform some malicious actions. For any processor, the set of instructions that consist only of alphanumeric characters is generally limited is size. Therefore it is non-trivial to construct shellcode that consists of only alphanumeric bytes. There exist a number of exploit tools that automatically translate non-alphanumeric shellcode into semantically equivalent alphanumeric shellcode for x86 architecture. To the best of our knowledge, there are no such tools available for ARM architecture. We report on our progress in developing a tool for automated generation of alphanumeric shellcode for ARM architecture.

1 Introduction

Errors introduced in writing programs can often be exploited to carry out malicious attacks on computer systems. For example, the goal of code injection attacks is to force a program to execute instructions that do not even appear in the original program. Typically, code injection attacks work in the following way. In the first step, the attacker sends the malicious data as input to the program, which then stores it in memory. The data is chosen by the attacker in such a way that it represents a valid machine code that performs malicious actions when executed. The usual goal of code injection attacks is to launch a command interpreter shell. Hence data input by the attacker is often referred to as *shellcode*. In the second step, the attacker exploits a vulnerability present in the program to divert the control to his shellcode. A wide variety of vulnerabilities can be exploited for this purpose, for example stack-based buffer overflows.

Most shellcodes in their original form consist of both alphanumeric and non-alphanumeric bytes. Thus, we may block data containing non-alphanumeric bytes to defend against code injection attacks. The challenge for the attacker is that of creating shellcode that consists of only alphanumeric bytes. Rix [2] was the first to discuss how to write alphanumeric shellcodes for x86 architecture. He developed a tool which takes non-alphanumeric code as input and outputs alphanumeric shellcode. The output generated is a self-modifying program. It consists of a decoder that re-constructs the original non-alphanumeric shellcode before executing it. The main disadvantage of his technique is the expansion in the size of the alphanumeric shellcode as compared to the input shellcode.

B. Gierlichs, S. Guilley, and D. Mukhopadhyay (Eds.): SPACE 2013, LNCS 8204, pp. 38–39, 2013.

In 2004 Berend Jan Wever [1] introduced the idea of loop decoding to help reduce the output size. The shellcode is encoded in such a way that the decoding can be performed iteratively, thus decreasing the size of the shellcode.

Younan and Philippaerts [4] showed that the subset of ARM machine code programs that (when interpreted as data) consist only of alphanumeric characters (i.e. letters and digits) is a Turing complete subset. A related work [3] showed that it is feasible to write alphanumeric shellcodes for ARM. The approach followed by these earlier works, however, is not amenable to automation as it requires alphanumeric shellcode to be crafted manually. We report on our progress in developing a tool for automated generation of alphanumeric shellcode for ARM architecture.

References

1. Wever, B.J.: Alphanumeric shellcode decoder loop. Skypher (2004)
2. Rix: Writing IA32 alphanumeric shellcodes. Phrack 57 (2001)
3. Younan, Y., Philippaerts, P.: Alphanumeric RISC ARM shellcode. Phrack 66 (2009)
4. Younan, Y., Philippaerts, P., Piessens, F., Piessens, F., Joosen, W., Lachmund, S., Walter, T.: Filter-resistant code injection on ARM. Journal of Computer Virology and Hacking Techniques 7(3), 173–188 (2010)