Runtime Process Infection

Shawn Webb, shawn@shawnwebb.info

September 18, 2012

Abstract

Injecting arbitrary executable code into a process is the end goal of exploitation. Many techniques already exist for injecting code. This paper will describe existing techniques as well as a novel new technique. The new technique provides stealthy injection into anonymous memory maps. The reader will need to be well-versed in ELF, Linux memory management on 32bit Intel systems, and the C programming language. Basic knowledge of assembly on 32bit Intel systems is assumed. Though the paper describes 32bit systems, a reference tool has been implemented that works on both 32bit and 64bit Linux systems.

1 Introduction

Writing malware on Windows is a relatively easy task. The win32 API is very friendly towards malware authors. Microsoft even provides a library called detours which makes injecting DLLs into a process during runtime extremely easy. Detours, however, does not provide stealth. Linux has a similar project called injectso. Injectso is a non-stealthy solution for injecting shared objects into a process during runtime.

If a system administrator thinks a service running on a server has been exploited, one thing the administrator can do is look at the `/proc/[pid]/maps` file and look at what shared objects are loaded. As an example, let's say we're targeting the Apache web server. We want to drop into a shell whenever the string "`GET /shell HTTP/1.1`" is sent to Apache over the socket. If we were to use injectso, we'd inject a shared object in such a way that the administrator would see an entry that matches `/path/to/evil.so`. Obviously, the system's administrator knows at that point the game is over. And so will you when you try to get back in.

Therefore, we need a stealthy way to inject our shared object. Our requirements are:

1. Inject shared objects
2. Hook dynamically-loaded functions
3. Stealth through anonymous injection

In the rest of this paper, we will discuss:

1. Current techniques for storing and executing arbitrary shellcode
2. The Executable and Linkable Format (ELF)
3. The new technique for storing and executing arbitrary shellcode
4. Hooking dynamically-loaded functions
5. Future work and research

2 Current Techniques

Many techniques exist for storing and executing arbitrary code. For decades, storing code on the stack has been a known technique[1]. Being well known, this technique is very well protected against[2].

[1] See aleph1's famous Phrack article, Smashing The Stack For Fun And Profit

[2] Add footnote for non-exec stack