Using Run-time Randomization Against Memory Corruption Attacks On Legacy

XI CHEN

Since first identified by Anderson in the 70s, memory corruption vulnerabilities are still among the top three of the CWE SANS most dangerous software errors. Even now, nearly 400 memory corruption errors have been discovered in CVEs in the past 10 years. Memory corruption errors are especially common in applications written in type-unsafe languages like C and C++, due to the pointer operations supported in these languages. The ability to manipulate pointers provides an efficient way to design applications and systems, but using it incorrectly often leads to unintentional modification of related memory content, enabling an attacker to exploit such a vulnerability by either compromising sensitive data or hijacking the control flow. It is hard to convince developers to switch to type-safe languages, not only because of performance, but also because of backward compatibility reasons.

The problem of improving the security of applications or systems written in C/C++ has been recognized by the community for a long time, but previous solutions often introduce non-acceptable overheads, and many require source information, which is often unavailable for legacy binaries.

In this dissertation, we argue that address space runtime randomization can be effective against memory corruption attacks with general applicability to different memory spaces: heap space, stack space and code space. Our investigation focuses on improving the security guarantees provided by ASLR and prior work in the area, while also improving performance, capabilities, and re-randomization frequency. To substantiate our claims, we present several contributions and demonstrate the viability of the proposed techniques in practice. In particular, we present the first self-contained solution to randomize legacy binaries at runtime without requiring for any hardware, kernel, or source code support.