Defense Against Android Native Code Exploitation

Yuan Tian, Yu Seung Kim, Chen Chen, and Patrick Tague

Background

- More apps include native code.
- An attacker can corrupt the memory of a vulnerable process by exploiting native code.
- Current defenses cannot prevent attackers from corrupting the memory.

Attacking Process

- An attacker inserts exploit code into stack, but the code cannot be executed since the stack is non-executable.
- An attacker overflows the program to call `mprotect()`, changing the stack executable.
- An attacker can now execute the inserted code.

Proposed Solution

- Whenever `mprotect()` is called to change memory block to be executable, check if requested block overlaps with any stacks.
- If yes, the request is denied and thus the code in the stack cannot be executed.

Implementation and Future Work

Current status

- Implemented in Android OS 4.2
- Compatible with legacy Android OS
- Does not break legitimate Apps

Next step

- Extend the protection to heap (need to consider Dalvik Just-in-time compilation)
- Develop a light-weight control flow integrity check