Blender

Self-randomizing Address Space Layout for Android Devices

Background: Security Mechanisms

- Since 1.5: ProPolice (Stack Protectors):
  - Random number between local variables and return pointer
  - Return is not executed if number is overwritten
- Since 2.3: DEP (Data Execution Prevention)/NX (Not eXecutable):
  - Memory pages are never marked as both executable and writable
- Since 4.0: ASLR (Address Space Layout Randomization):
  - base addresses of stack, heap, system and dynamic libraries are randomized
- Since 4.1: PIE (Position Independent Executable)
  and RELRO (Relocation Read-Only)

Background: Recap: ROP

- whenever a function is called, the later next instruction is pushed on the stack
- if there is a buffer overflow you can overwrite this value (return pointer)
- this makes it possible to hijack the programs control flow
- the attacker can chain together many addresses on the stack (ROP chain)
- these addresses are called ROP-gadgets and together make a new program logic

Background: Android Attack Surfaces

- weakened ASLR:
  - the zygote process forks itself for every started app, memory layout is inherited
  - therefore memory layout is shared between all running apps and predictable
- ART vs. DalvikVM:
  - ART (Android RunTime) as the successor of the DalvikVM
  - the ART loads well defined native API code into the memory
  - base address of the ART code section is not randomized sufficiently
- malicious apps:
  - a malicious app could read the shared memory layout, stack cookie secrets etc.
  - this can happen with full authorization of the user
- high number of ROP-gadgets:
  - preloaded libraries, ART

Blender: Structure

- Blender bootstrap module
  - takes over startup of the app, invokes other modules
- Blinker (Blender dynamic linker)
  - rearranges preloaded libraries and loads other libraries to randomized addresses
- BlenderLRM (Blender Library Randomization Module)
  - organizes rearrangement of preloaded libraries
- BlenderART (Blender ART Randomization Module)
  - rearranges the ART native code to a randomized address

Blender: Implementation - BlenderLRM

- most system libraries are dynamically linked
- linking happens with the creation of the zygote process
- dependencies between libraries -> no simple relocation
- computation of dependency graph
- relocate library and fix all references to all GOT entries of the library

Blender: Implementation - BlenderART

- fix all absolute addresses before relocation:
    - find all absolute addresses with Google’s “oat_patch” tool
    - rewrite addresses for all found patches
    - patch metadata of the oat-header and section headers
- fix the Class Linker Data Instance
    - method tables also contain absolute addresses
- mark the old memory region as non-executable
    - cannot be fully unmapped because there are still absolute data-pointers

Blender: Performance Evaluation

- high increase in average memory entropy:
  - 0.005 vs 0.991 for original app vs full Blender
- increases startup overhead noticeably:
  - increases startup time by almost one second
  - only affects (cold) startup, not runtime
  - highly optimizable with pool of pre-relocated libraries
- negligible memory and battery consumption overhead

Conclusion

- the zygote app creation process weakens ASLR on android
- together with the new ART this creates many unnecessary threats
- the methods proposed in this paper could mitigate them effectively