Return Oriented Programming - Why?

Sanjay Rawat

Why ROP?

Why ROP?

- Inability to predict accurately whether a particular execution will be benign or not.

Why ROP?

- Inability to predict accurately whether a particular execution will be benign or not.
- Instead focused on preventing the introduction and execution of new malicious code.

Why ROP?

- Inability to predict accurately whether a particular execution will be benign or not.
- Instead focused on preventing the introduction and execution of new malicious code.
- Two directions:
  - Control flow integrity proof
  - Isolate “bad” code that has been introduced into the system.

W⊕X feature

- Memory is either marked as writable or executable, but may not be both.
- Prevents the execution of shellcode, even if we are able to bypass CFI and able to write the shellcode .
- Intel and AMD offer this feature and operating systems- Windows Vista, Mac OS X, Linux, and OpenBSD now support.

Where is the problem

- Flawed assumption: preventing the introduction of *malicious code* is sufficient to prevent the introduction of *malicious computation.*
- *Return oriented programming* is a proof of this flawed assumtion.
- We'll get acquainted with ROP shortly!!

X86 and ROP

- instruction set is large and its encoding is dense => a variety of instructions are available for use even in relatively small programs.
- calling convention uses the stack, which an attacker can often overwrite (something belongs to the attacker!)
- ROP Principle: How should programs be constructed if the stack pointer takes the place of the instruction pointer?

Reference

- *Return-Oriented Programming: Systems, Languages, and Applications* By RYAN ROEMER, ERIK BUCHANAN, HOVAV SHACHAM and STEFAN SAVAGE
- For working example:
  - Return Oriented Programming and ROPgadget tool by Jonathan Salwan
  - http://shell-storm.org/blog/Return-Oriented-Programming-and-ROPgadget-tool/