ROSS: Randomization of Operating System for Security

Anya Berdichevskaya
Jared Crane
Wenliang Du (Kevin)
Rajeev Gopalakrishna
Advisor: Gene Spafford

Problem

- An attack script developed on one machine is likely to work on thousands of other machines.
- Examples:
  - Denial-of-service attack.
  - Buffer overflow attack.
  - Virus, worm.

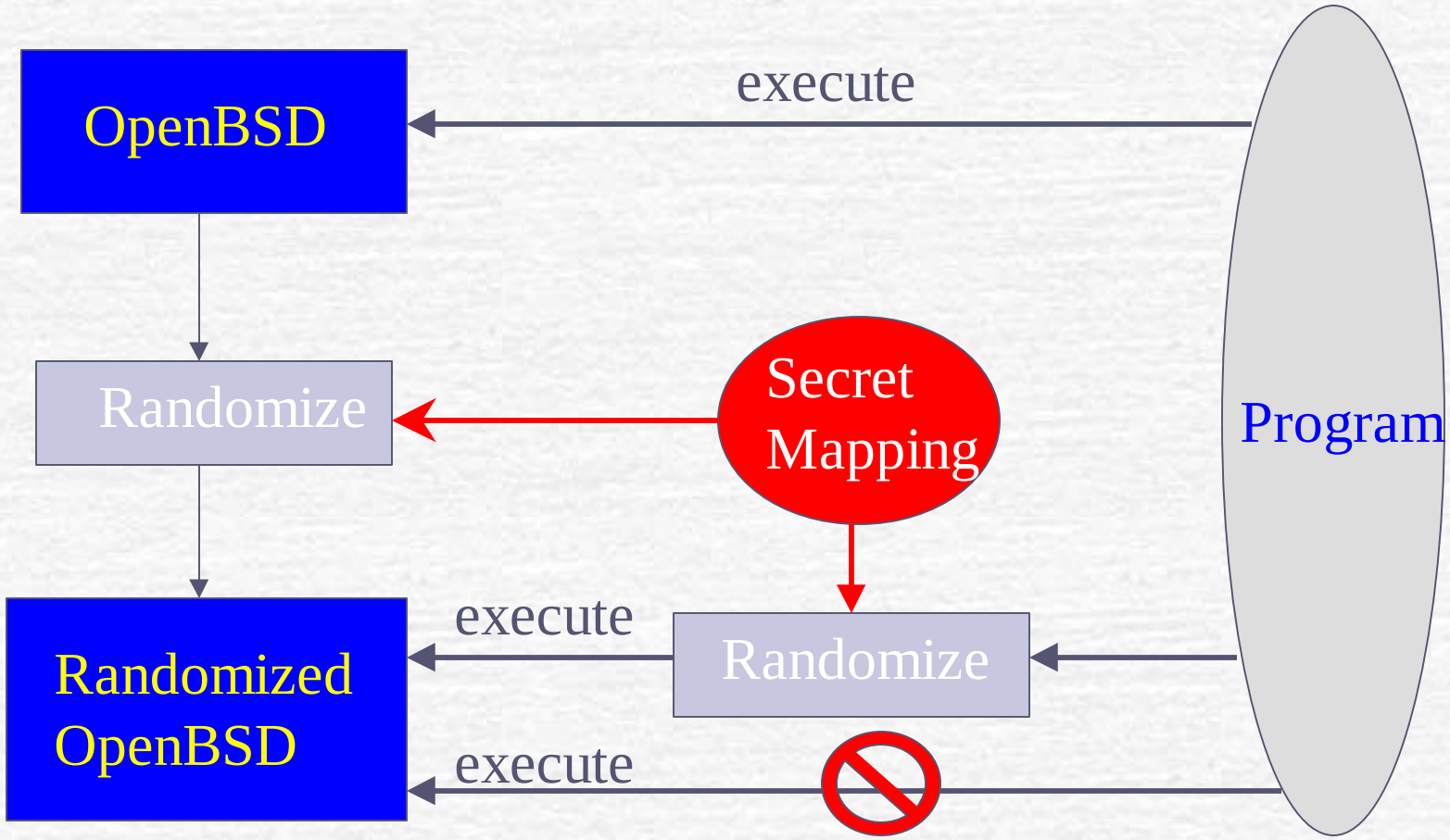
Motivations

- If every intrusion, virus, or worm had to be crafted explicitly to a particular machine, the cost of trying to penetrate computer systems would go up dramatically.
- How to do this?
  - Randomization or specialization of software.

Objectives

- Randomization on operating system.
  - What features of an operating system can be randomized to improve system security, and how?
  - How does that affect usability?
  - How does it affect system performance?
- Develop a prototype system based on OpenBSD as a proof of concept.

Framework


OpenBSD
execute
Randomize
Secret
Mapping
Program
Randomized
OpenBSD
execute
Randomize
execute

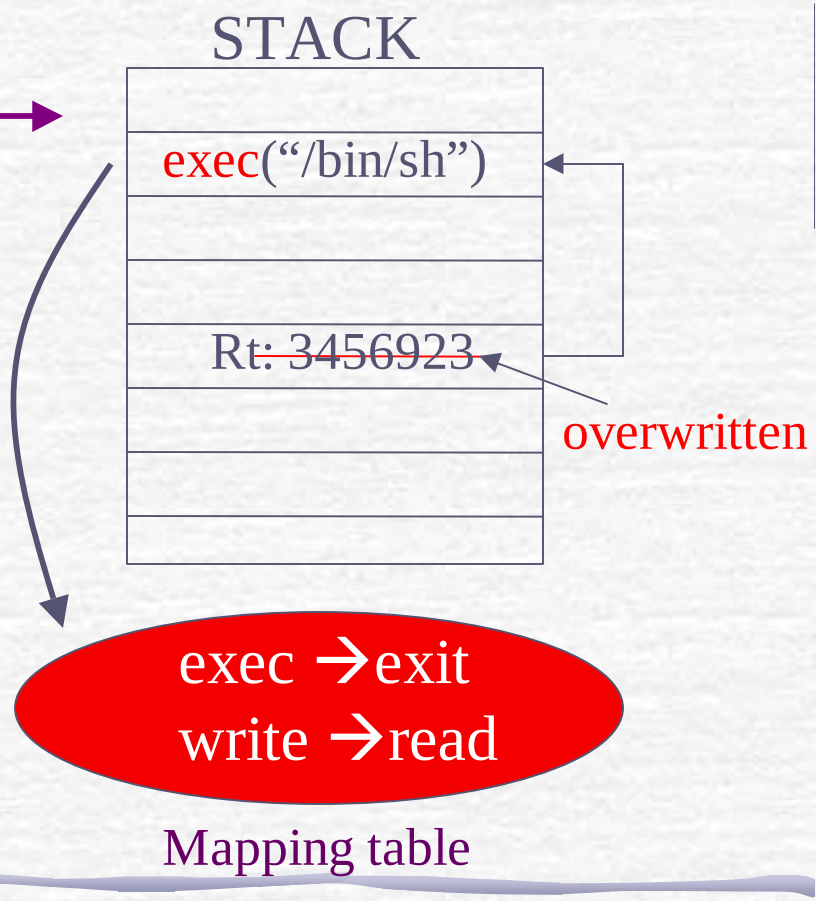
What Can Be Randomized?

- Dynamic libraries.
- System calls.
- Names for system files.
- Environment variables.
- Data locations.
- System behaviors.
- And more …

Current Work

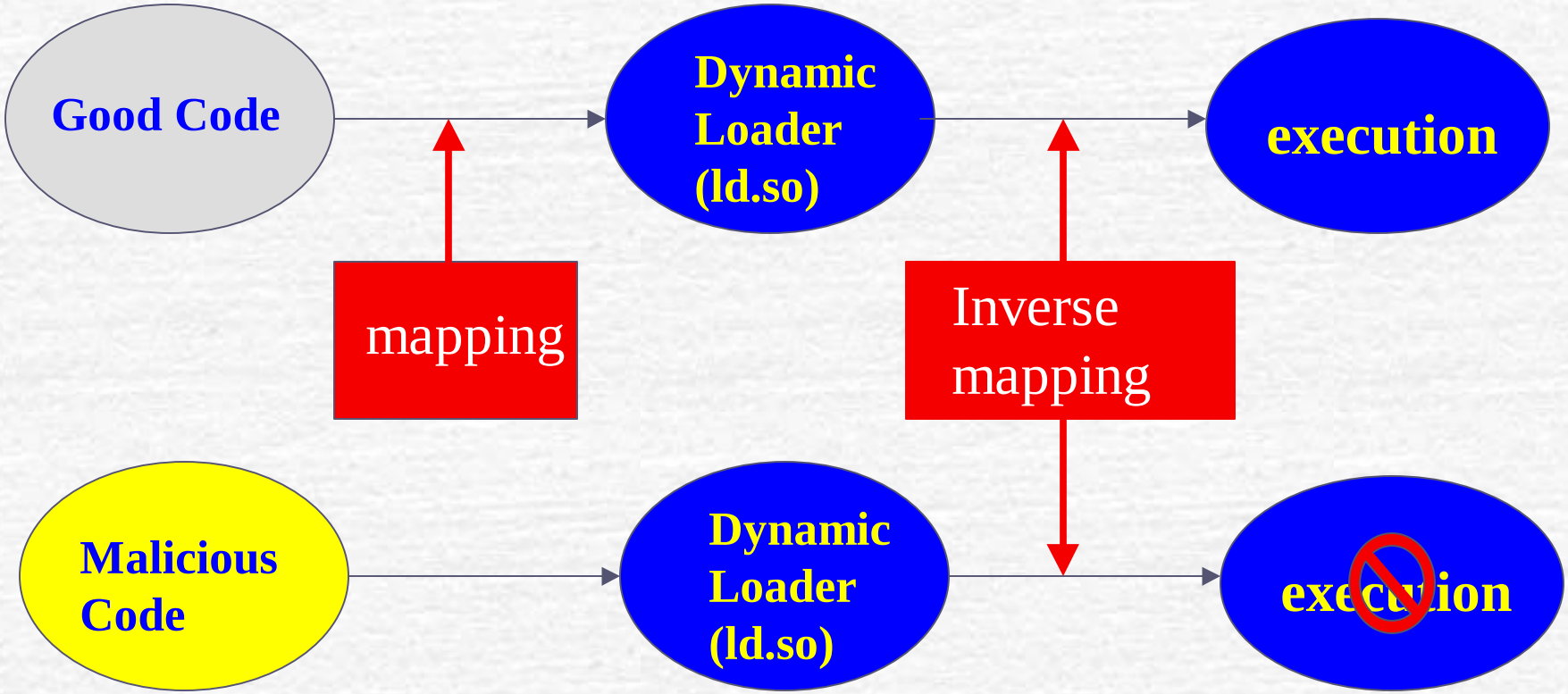
- Randomizing system calls.
  - Write() → exit(), execve() → kill();
- Randomizing dynamic libraries.
  - Dynamic loader: ld.So.
  - fwrite() → fread(), fread() → fclose();

Example 1: Buffer Overflow



correct
execution

Example 2: Malicious Code



Limitations

- What it works for:
  - Specialized machine, such as firewall
  - Relatively stable environments
- What it does not work for:
  - Inside intrusions
  - Dynamic environments (inconvenience)

Future Plans

- Investigate other features.
- Study specialization techniques used for other purposes, and try to use them for the security purpose.
- Randomizing applications.
  - How to systematically randomize applications?