Fine-Grained Load-Time Randomization of ELF binaries

Parsa Sadri Sinaki, Arto Niemi

- **70%** of compiled code **vulnerabilities** are due to **memory safety** issues.
- Return-oriented programming (ROP) is a class of code reuse attacks that attackers are widely using.
- **Fine grained code randomization** is a **solution** to mitigate this type of code reuse attacks.

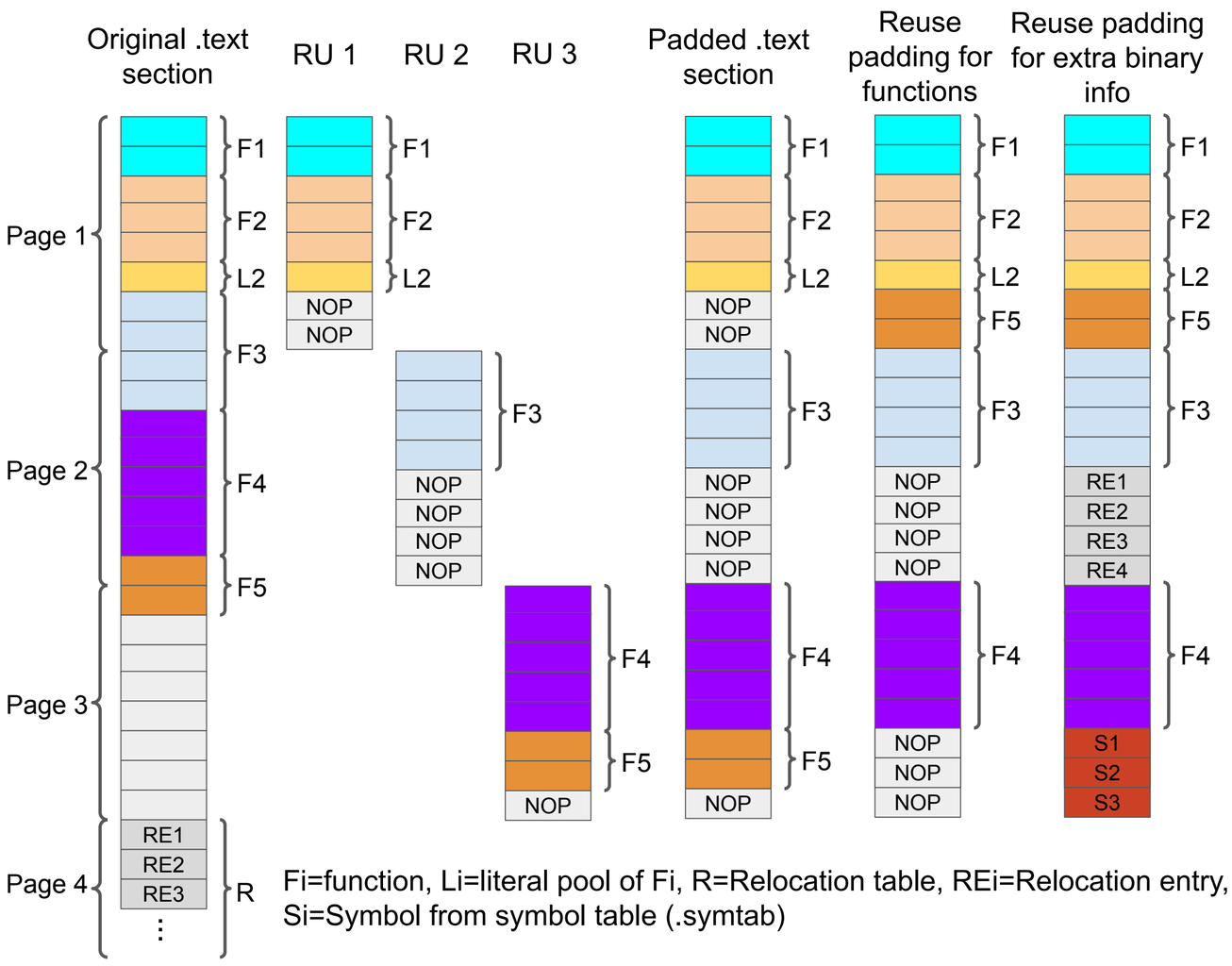
Randomization requirements

- Randomization per execution to be secure against brute forcing
- Randomization at **load time** or later to be secure against static analysis
- **Randomize each page when it is loading into memory** (page fault)
  - Reduce load time overhead
  - Not randomize pages that will not be loaded

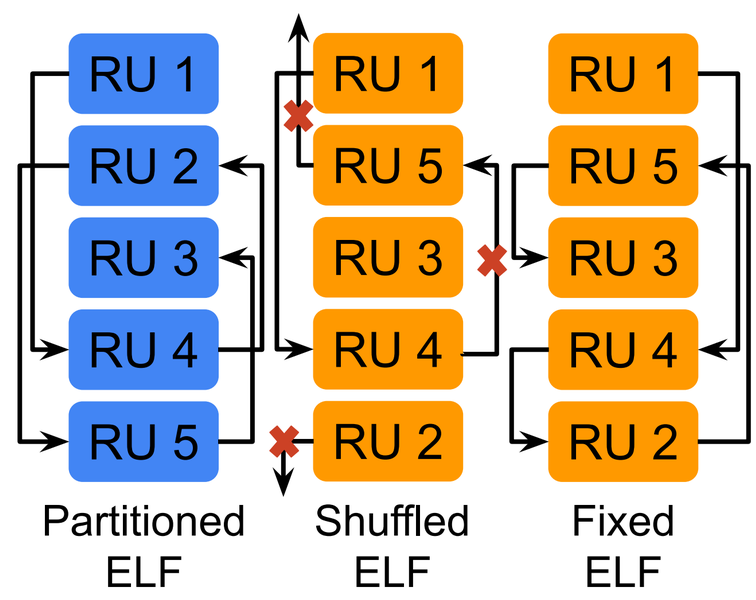
- Exploiting a memory safety vulnerability:
  - Overflow (Mitigated by control-flow integrity and stack canaries)
  - Code injection (Mitigated by W⊕X)
  - Code reuse (Mitigated by ASLR)
  - **Code reuse + information disclosure** (code randomization) (No widely deployed method)

ELF randomization steps

1. **Partition** code section into Randomization units (RU) with configurable size (e.g. page size)
2. **Shuffle** the RUs randomly
3. **Fix** all **addresses** used in the binary (e.g. target in branch instructions)



Fi=function, Li=literal pool of Fi, R=Relocation table, REi=Relocation entry, Si=Symbol from symbol table (.symtab)



Problem

- **Functions** in an RU need to **not cross page boundary** so randomization doesn't break functions

Solution

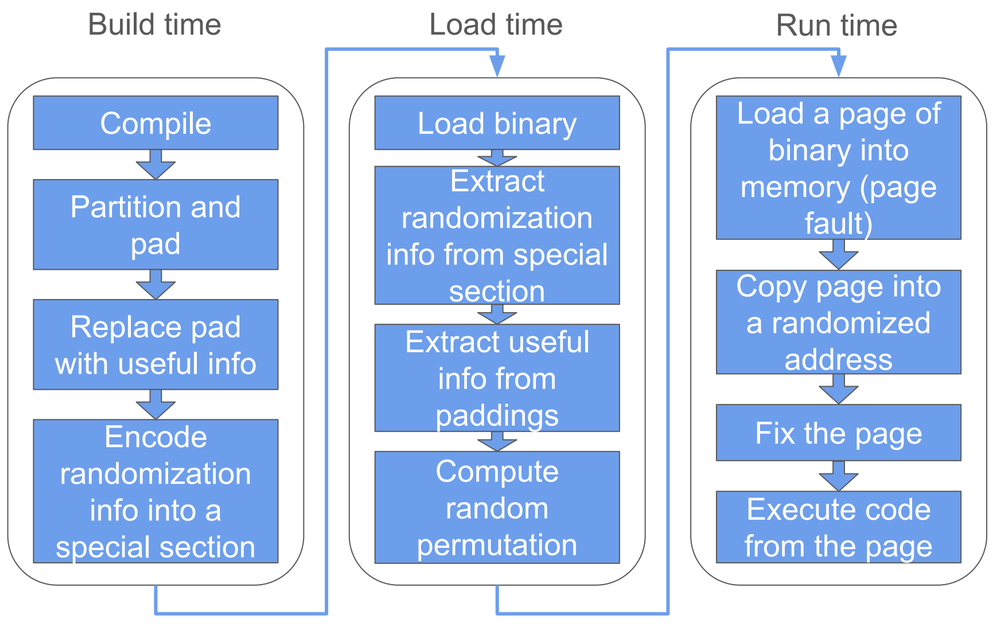
- **Insert padding** after each RU to **make page aligned**

New problem

- Padding at the end of RUs, **expands** the **original binary**

New solution

- **Repurpose paddings** to store randomization data, relocation entries, or RUs.



Conclusion

- **Defense** against **code reuse attacks (ROP)** in presence of **information disclosure**
- **Reduce** the **size** of **randomized binaries**

Future work

- **Randomise** pages in demand on **load time** (modify loader)
- **Compatibility** with **signature** and attestation
- **Compatibility** with **page sharing** in shared libraries

Helsinki System Security Lab (HSSL)

HSSL drives renewal and mastery in the field of platform and device related security technologies, especially for Huawei consumer devices such as mobile phones, laptops, televisions and automotive. We do research in topics such as hardware-assisted isolation and integrity, as well as in operating system protection (hypervisor, TEE, secure enclaves and kernel hardening). We also carry expertise in cryptography and systems security functionality such as device key management (PKI), device attestation and key-store solutions.

HUAWEI