**Survey Motivation**

**Binary Rewriting?**
- Software is often distributed in binary form or needs to be changed during runtime.
- Originally inspired by the need to change parts of a program while software is executed.
- Nowadays, evolved into a plethora of approaches with different application domains (e.g., Emulation, Observation, Optimization, Hardening).

**Problem**
- A plethora of different approaches and methods has led to the development of many different tools.
- However, because of this, it is not always easy to identify the right tool for the problem at hand.
- Additionally, the availability of tools and methods for specific purposes is not well studied.

**Rewriting at a Glance**

**Transformations**

- **Static**: perform alterations directly at instrumentation point (e.g., during link time)
- **Dynamic**: Able to perform changes at instruction granularity during runtime
- **Minimal-invasive**: operations on branch granularity, by redirecting control flow to newly generated code
- **Full-translation**: transform binaries at any instruction, but require lifting into Intermediate Representation (IR)

**4 Steps of Rewriting**

1. **Parsing**: Extract instruction and data stream from binary objects for further analysis.
2. **Analysis**: Provides information on building blocks (e.g., disassembly, structural recovery or label, symbol and data type extraction)
3. **Transformation**: Prepare instrumentation points and define alterations (e.g., to instructions or control flow)
4. **Code Generation**: Apply the intended changes into the binary of interest in a way to keep it executable

**Conclusion**

- Full-translation-based schemes allow for application of reasoning approaches due to the more abstract representation of the binary under investigation.
- Currently only semantic equivalent lifters are available, which are sufficient for many applications.
- Scenarios like altering timing sensitive applications, performance optimization for throughput-oriented programs, or rewriting software with real-time requirements would greatly benefit from instruction equivalent lifters.
- The x86 architecture is still the primary target for binary rewriting applications, but other architectures like ARM and MIPS draw more and more interest.

---