In future, we want to further improve the Methodology. The class hierarchy analysis (CHA) is used to compute virtual table layouts such that we obtain the smallest possible range for each indirect call site. We interleave virtual table layouts such that we obtain the smallest range candidate per call-site. We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.

2. Contributions

- We reduced the number of call-targets per call-site, thus improving the precision of our mapping. (precision)
- We decreased the performance overhead w.r.t. [1]. (performance)
- We decreased the binary blow-up size. (binary size)
- We improved the protection coverage. (increased security level)

3. Motivating Example

Consider: Base* obj2 = new Base(); obj2 = vfn();

The virtual pointer can be corrupted (i.e., red arrow from above Figure) to point into a different virtual table. The new virtual table is (not) in the expected class or virtual table hierarchy.

4. Background

5. Design

- Obtain the object type and the virtual table used during the object dispatch.
- We interleave virtual table layouts such that we obtain the smallest possible range for each indirect call site.
- We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.

6. Implementation

- The Clang (LLVM front-end) is extended in order to provide the virtual tables as metadata during LLVM link time.
- The class hierarchy analysis (CHA) is used to compute virtual table inheritance paths.
- The virtual table inheritance paths are analyzed in order to derive permissible and impermissible ranges for each call-site.
- The new range checks are added before each indirect call site.

3. Motivating Example

Consider: Base* obj2 = new Base(); obj2 = vfn();

The virtual pointer can be corrupted (i.e., red arrow from above Figure) to point into a different virtual table. The new virtual table is (not) in the expected class or virtual table hierarchy.

4. Background

5. Design

- Obtain the object type and the virtual table used during the object dispatch.
- We interleave virtual table layouts such that we obtain the smallest possible range for each indirect call site.
- We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.

6. Implementation

- The Clang (LLVM front-end) is extended in order to provide the virtual tables as metadata during LLVM link time.
- The class hierarchy analysis (CHA) is used to compute virtual table inheritance paths.
- The virtual table inheritance paths are analyzed in order to derive permissible and impermissible ranges for each call-site.
- The new range checks are added before each indirect call site.

3. Motivating Example

Consider: Base* obj2 = new Base(); obj2 = vfn();

The virtual pointer can be corrupted (i.e., red arrow from above Figure) to point into a different virtual table. The new virtual table is (not) in the expected class or virtual table hierarchy.

4. Background

5. Design

- Obtain the object type and the virtual table used during the object dispatch.
- We interleave virtual table layouts such that we obtain the smallest possible range for each indirect call site.
- We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.

6. Implementation

- The Clang (LLVM front-end) is extended in order to provide the virtual tables as metadata during LLVM link time.
- The class hierarchy analysis (CHA) is used to compute virtual table inheritance paths.
- The virtual table inheritance paths are analyzed in order to derive permissible and impermissible ranges for each call-site.
- The new range checks are added before each indirect call site.

3. Motivating Example

Consider: Base* obj2 = new Base(); obj2 = vfn();

The virtual pointer can be corrupted (i.e., red arrow from above Figure) to point into a different virtual table. The new virtual table is (not) in the expected class or virtual table hierarchy.

4. Background

5. Design

- Obtain the object type and the virtual table used during the object dispatch.
- We interleave virtual table layouts such that we obtain the smallest possible range for each indirect call site.
- We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.

6. Implementation

- The Clang (LLVM front-end) is extended in order to provide the virtual tables as metadata during LLVM link time.
- The class hierarchy analysis (CHA) is used to compute virtual table inheritance paths.
- The virtual table inheritance paths are analyzed in order to derive permissible and impermissible ranges for each call-site.
- The new range checks are added before each indirect call site.

3. Motivating Example

Consider: Base* obj2 = new Base(); obj2 = vfn();

The virtual pointer can be corrupted (i.e., red arrow from above Figure) to point into a different virtual table. The new virtual table is (not) in the expected class or virtual table hierarchy.

4. Background

5. Design

- Obtain the object type and the virtual table used during the object dispatch.
- We interleave virtual table layouts such that we obtain the smallest possible range for each indirect call site.
- We filter the resulted ranges based on virtual table inheritance paths such that we obtain the smallest candidate range per call-site.