Type Flattening Obfuscation

Ta Thanh Dinh
tathanhdinh@gmail.com

***Abstract*—Beside data and control flow, high-level types are important in binary code analysis, particularly in decompilation. Some research papers have introduced methods to map machine-dependent objects into types of some C-like type system. For the anti-decompilation/obfuscation purpose, we present a novel technique which bypasses existing type recovery approaches. We have implemented a proof-of-concept obfuscating C compiler to demonstrate the technique, the compiler is given open source.**

***Index Terms*—type recovery, decompilation, obfuscation**

1. Introduction

Binary code *decompilation* [1] is to transform the low-level, machine-dependent code of a program into a high-level form, like code of a high-level language. In almost all academic research papers and commerical products, the target language is C. Similar to compilers, a modern binary code decompiler consists of many phases [1, 5]: disassembly, function boundary detection, immediate representation (IR) lifting, control-flow graph (CFG) recovery, high-level variables detection, type (i.e. variable types and function signatures) recovery, etc. Each phase requires particular, but not independent [3], analysis techniques; the analyzed program is transformed gradually into a higher-level, more abstract and more understandable representation.

In the opposite direction, binary code *obfuscation* is a method to protect the low-level code from being decompiled, or from being analyzed in general. Because the code analysis contains of different interdependent phases, the obfuscation [6, 13] can proceed at any of them, e.g. anti-disassembly (binary packer, self-modifying code), binary stripping, control-flow flattening, virtualization (for both data and control obfuscation)... just name a few. Basically, each obfuscation method consists of one or several *semantics-preserving* transformations [4, 6] which hide certain properties of the code.

An optional feature of binary code decompilation is *type reconstruction*, namely to recover high-level types from machine-dependent objects [2, 5]. This is the research objective of some research papers [8, 9, 11, 12], and killing feature of some commercial [15, 16] as well as open source [14] binary code analysis tools. Beside decompilation, types and particularly *function signatures* are also essential in numerous applications, e.g. static binary instrumentation [7], see for example [10] for a more complete list. The knowledge about types then expand the attack surface to programs need to be protected.

Despite of successes in type reconstruction and the need of protecting function signatures, to the best of our knowledge there is no explicit effort in type obfuscation. This paper then proposes a method for type obfuscation, the principal ideas are that the compiler does not need to preserve all information about high-level types (type erasure) and the *semantics gap* between the high-level language and machine code can be exploited to make some information very hard if not impossible to be recovered. We do not claim that all type information can be hidden, the attacker can eventually know some but it is not possible to distinguish the concrete underlying type from one to another, thus the notion of *type flattening*.

2. Ease of Use

2.1. Maintaining the Integrity of the Specifications

The IEEEtran class file is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

3. Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections 3.1–3.5 below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not number text heads—LaTeX will do that for you.

3.1. Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, ac, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

3.2. Units

- Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as "3.5-inch disk drive".

- Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
- Do not mix complete spellings and abbreviations of units: "Wb/m^2" or "webers per square meter", not "webers/m^2". Spell out units when they appear in text: ". . . a few henries", not ". . . a few H".
- Use a zero before decimal points: "0.25", not ".25". Use "cm^3", not "cc".)

3.3. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

3.4. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

3.5. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**b7**].

3.6. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

3.7. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

3.8. Figures and Tables

Positioning Figures and Tables. Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 1”, even at the beginning of a sentence.

TABLE 1. TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

Figure 1. Example of a figure caption.

Acknowledgment

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

References

[1] C. Cifuentes. “Reverse Compilation Techniques”. PhD thesis. 1994.

[2] A. Mycroft. “Type-Based Decompilation (or Program Reconstruction via Type Reconstruction)”. In: *ESOP*. 1999.

[3] B. Schwarz, S. Debray, and G. Andrews. “Disassembly of Executable Code Revisited”. In: *WCRE*. 2002.

[4] M. Dalla Preda and R. Giacobazzi. “Semantic-Based Code Obfuscation by Abstract Interpretation”. In: *ICALP*. 2005.

[5] M. Van Emmerik. “Static Single Assignment for Decompilation”. PhD thesis. 2007.

[6] C. Collberg and J. Nagra. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*. 1st. Addison-Wesley Professional, 2009.

[7] A. R. Bernat and B. P. Miller. “Anywhere, Any-Time Binary Instrumentation”. In: *PASTE*. 2011.

[8] J. Lee, T. Avgerinos, and D. Brumley. “TIE: Principled Reverse Engineering of Types in Binary Programs”. In: *NDSS*. 2011.

[9] K. ElWazeer, K. Anand, A. Kotha, M. Smithson, and R. Barua. “Scalable Variable and Data Type Detection in a Binary Rewriter”. In: *PLDI*. 2013.

[10] J. Caballero and Z. Lin. “Type Inference on Executables”. In: *ACM Computing Surveys* 48.4 (2016).

[11] M. Noonan, A. Loginov, and D. Cok. “Polymorphic Type Inference for Machine Code”. In: *PLDI*. 2016.

[12] E. Robbins, A. King, and T. Schrijvers. “From MinX to MinC: Semantics-Driven Decompilation of Recursive Datatypes”. In: *POPL*. 2016.

[13] S. Banescu and A. Pretschner. “A Tutorial on Software Obfuscation”. In: *Advances in Computers*. Vol. 108. Elsevier, Jan. 2018, pp. 283–353.

[14] *Ghidra*. URL: https://ghidra-sre.org/.

[15] *Hex-Rays Decompiler*. URL: https://www.hex-rays.com/.

[16] *JEB Decompiler*. URL: https://www.pnfsoftware.com/.