Type Flattening Obfuscation

Ta Thanh Dinh

tathanhdinh@gmail.com

Abstract. Beside data and control flow, high-level types are also important in program analysis, indeed type reconstruction is an essential step in binary code decompilation. For the purpose of anti-decompilation, the paper introduces a novel obfuscation technique which makes types become harder to be reconstructed.

1 Introduction

An important step in binary code *decompilation* [1] is to detect high-level objects (e.g. functions, variables) from low-level machine dependent objects [2] (e.g. registers, raw memory accesses, machine instructions) before annotate them with types of the supposed original language. Because most of information about high-level types is lost in compilation, the binary code *type reconstruction* requires special techniques, a survey up until 2015 can be referenced in [4].

In the opposite direction, binary code *obfuscation* is to make the decompilation (or in general code analysis) harder. Since the data and control flow are principal elements for the program analysis, current obfuscation techniques [3] focus mostly on hiding them. But to the best of our knowledge, there is still no explicit effort in hiding high-level types. There would be several reasons for this lack of effort.

First, type obfuscation can be an indirect effect of data or control flow obfuscation. Indeed, type reconstruction algorithms need both data and control flow to build type constraints, if any of them is hidden then the algorithms will not work. Or if the function boundary cannot be found (because of anti-disassembing tricks, for example), then there are no inputs (i.e. high-level objects) for type reconstruction. *Second*, high-level types seem too coarse to be worthy of being protected, in many cases just knowing certain values of the input which make the program exploitable is enough. But knowledge about types enlarges attack surfaces because more analysis can be proceeded, beside decompilation see the survey [4] for a more complete list.

2 Type flattening compiler

We introduce *ucc*, a proof-of-concept C compiler that explicitly obfuscates high-level types. We attacks on the core of type reconstruction algorithms, that is the data flow used to build type constraints, and the *semantic gap* between types in the high-level language and their representations in low-level machine code.

The compiler is open source and implemented in Rust. The source code, an user guide as well as some demo results are available at (cite).

References

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

[2] G. Balakrishnan and T. Reps. "DIVINE: DIscovering Variables IN Executables". In: *VMCAI*. 2007.

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

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