TaintBlade: A framework for automated protocol reverse engineering and study of botnet command and control protocols

1st Given Name Surname
dept. name of organizatiosn (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—TODO redact this at the end**

***Index Terms*—TODO keywords**

I. Introduction

TODO - This section introduces the problem we are dealing with and motivates the creation of this tool

II. Background

TODO - This section goes over the state of the art on analyzing malware binaries, past work on protocol reverse engineering and available tools.

III. Design

TODO - This section covers the general structure of the tool and the different modules and functionalities implemented.

- Requirements: overview and goals of the tool - Arch: overview and details of each stage - Implementation, with repo and how to use

In the previous sections, we introduced the issue of automating protocol reverse engineering and the relevance of this task in the context of the analysis of malware transmitting malicious communications. We will now describe *TaintBlade*, the automated protocol reverse engineering framework we have developed for tackling this task. Overall, this tool pursues the following three main goals: (1) to perform message inference of the underlying ingress protocol of a program, offering an accurate representation of its compounding fields and the purpose of each byte; (2) to provide a complete trace of the activities carried out by the instrumented program, including a viewpoint into loaded images, child process, executed routines and their arguments, therefore helping the analyst with the reversing task, instead of becoming a black-box tool; (3) to become an usable framework not only by malware analysts by offering specific functionalities aimed for studying malware behaviours, but also by the general public by providing an intuitive and fully enriched graphical user interface.

IV. Evaluation

TODO - This section covers multiple test cases, including custom programs that we will include in our repository that showcase some easy-to-understand cases we want to show that work, and some others that are real malware found in the wild.

Goal of the eval first. It's about checking functionality, and checking that we detect e.g. all commands we know are at the sample

V. Related work

TODO decide whether to include this or not - It might fit better in the background

VI. Conclusion

TODO - This section offers an overview of the developed tool and the results obtained over the problem we were trying to solve.

A. Future work & limitations

Acknowledgment

TODO

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "Sample title," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.