coggle
made for free at coggle.it

2017 - Assisting Malware Analysis with Symbolic Execution: A Case Study

- Goals
  - Show how an analyst can use symbolic execution techniques to unveil critical behavior of a remote access trojan
- Authors
  - D'Elia D C
  - Baldoni R
  - Coppa E
  - Demetrescu C
- Information
  - Malware is responsible for the most frequent and costly attacks on public and private organizations
  - Case study
    - RAT
      - Remote access trojan
      - Drawn from a family of backdoor malware specifically created to execute remote commands in Microsoft Windows platforms
      - Gathers information and communicates with a command-and-control server.
      - Once activated, the malware allows a remote attacker to take over the infected machine by exchanging files with the server and executing shell commands.
      - It then copies itself in a number of executable files of the Windows system folder and modifies the registry so that it is automatically launched at every startup.
      - Malware uses thread injection to activate its payload in Windows Explorer. The payload connects to http://mse.vmnat.com, sending a sprintfformatted string with information on the system retrieved using the NetbiosAPI
      - Core of the malware is a loop that periodically polls the server for encrypted remote commands and decrypts them by applying a character-wise XOR with the 0x45 constant
  - Feature
    - Automatically explores the possible execution paths of bounded length starting from a given entry point. The analysis is static and the code is not concretely executed
    - As output, the tool produces a report that lists for each explored execution path the sequence of encountered API calls and their arguments, along with properties of the malware's input for which the path is traversed,
    - e.g., the actual data values read from a socket that would trigger the path's execution.
  - Symbolic execution
    - Analysis technique for testing a property in program against multiple execution paths at a time
    - Program is allowed to take on symbolic rather than concrete input values, while an execution engine collects across each explored path a set of constraints that are combined into a formula describing the path
    - When an assignment instruction is evaluated, the formula is simply updated to reflect it.
    - When a branching instruction is encountered and its outcome depends on one or more symbolic values, the execution is forked by creating two states described by two distinct formulas, derived from the current formula by adding to it the branch condition or its negation, respectively.
    - A constraint solver - typically one suited for satisfiability modulo theories (SMT) - is used to evaluate expressions involving symbolic values, as well as for generating concrete inputs that can be used to run concretely the program along the desired path
    - Challanges on Malware Domain
      - Majority of currently available symbolic executors are not well equipped for analyses in the malware realm
      - Path explosion problem
        - As a symbolic executor may fork states at every input dependent branch, the total number of paths to explore might be exponential.This impacts both space and time
        - Common approach is to employ search strategies that can limit the exploration to a subset of paths that look appealing for a given goal
        - Domain-specific optimizations and search heuristics, can mitigate the path explosion problem in the analysis of malicious binaries, making their symbolic analysis feasible
    - Dissecting RAT with angr
      - How the dissection of the RAT sample can be carried out
      - Execution Context
      - Starting the Exploration
    - RAT Dissected
  - angr
    - Symbolic executor
    - Extensions
      - A number of extensions to the original angr framework is devised to make dissection of RAT possible
      - Implement 57 models of commonly used function, model is a summary for a function that simulates its effects by propagating symbolic data in the same way that the original function would have, requiring a significantly shorter amount of time than in a symbolic exploration.
  - What is?
- Contribution
  - Demonstrate how symbolic execution can be applied to malware analysis