Detecting Process Memory Tampering

Bachelor thesis

Supervisors: Prof. Dr. Alexander Pretschner, Sebastian Banescu
Email: {alexander.pretschner, sebastian.banescu} @ in.tum.de
Phone: +49 89 289 – 17, 314
Starting date: immediately

Context
Modern operating systems are able to detect tampering of software binaries (i.e. binary patching) via digital signatures [1]. For instance, Microsoft Windows checks if installer executables are signed by a verified publisher and notifies the user about this publisher. However, this defense does not protect against process memory tampering (e.g. DLL injection and remote thread creating), which compromise the integrity of applications, leading to undesired behavior [2].

Process memory tampering has long been problem for massive-multiplayer online gaming vendors, because cheaters generally employ it to obtain an advantage over other players [3]. This method is used in first-person shooter games to build aim-bots, see though walls, get unlimited ammo, defy gravity, etc. However, process memory tampering is also employed by malware running at non-administrative user level, in order to change the behavior of applications to the advantage of the malware writer. One example is browser-hijacking where the malware changes the web-browser behavior by injecting unwanted advertisements into search-results and changes the preferred settings of the user [4].

Goal
The goal of this thesis is to design and implement a process memory tampering detection mechanism. The mechanism can be based on software runtime process invariants (e.g. memory contents and behavioral characteristics), described in the literature [3] and/or designed by the student.

A concrete case-study will be performed using the Chrome/Chromium web-browser. An evaluation of the implementation will be performed with a set of malicious applications which tamper with process memory of the target web-browsers.

Work-plan
1. Develop knowledge of process memory checking techniques:
   a. Read references [3], [4], [5], [6] and find related work on this topic
   b. Write a state-of-the-art survey, which presents and compares the investigated techniques.
2. Design and implement the process memory checking mechanism:
   a. Choose technique(s) described in literature and/or propose a new technique; argument your choice (e.g. security versus cost tradeoff) in written form.
   b. Implement the chosen technique(s) and document design decisions
3. Evaluation of the implementation (case-study):
   a. Select a set of common malware samples which tamper with process memory of web-browsers. Or design a toy malware with this functionality (e.g. this malware could use the MS UI Automation API to perform unwanted / malicious actions on the target program).
   b. Measure effectiveness of implementation against the selected malware.
c. Measure performance impact of the implementation on the operating system and on the target application, i.e. the web-browser.

d. Analyze and discuss security versus performance issues of the implemented solution.

4. The final thesis document must contain:
   a. Description of the problem and motivation for the chosen approach
   b. State of the art survey, including analysis of security and performance
   c. Rationale for choosing certain technique(s) for implementation
   d. Implementation description
   e. Performance evaluation of implementation
   f. Discussion on potential security and performance threats
   g. Conclusions and future work.

Deliverables
- Virtual machine able to run a demo of the implementation, including instructions on how to run the demo.
- The VM should also include the source code of the implementation.
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions.
- Final thesis report written in conformance with TUM guidelines.

References