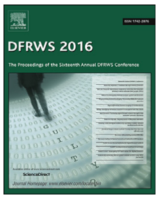
Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



DFRWS USA 2016 — Proceedings of the 16th Annual USA Digital Forensics Research Conference

Robust bootstrapping memory analysis against anti-forensics

Kyoungho Lee [a], Hyunuk Hwang [b, *], Kibom Kim [b], BongNam Noh [a]

[a] *System Security Research Center, Chonnam National University, Gwangju, South Korea*
[b] *The Affiliated Institute of ETRI, Daejeon, South Korea*

Keywords:
Windows
Memory analysis
Memory forensics
Robust analysis
OS fingerprinting

A B S T R A C T

Memory analysis is increasingly used to collect digital evidence in incident response. With the fast growth in memory analysis, however, anti-forensic techniques appear to prevent it from performing the bootstrapping steps — operating system (OS) fingerprinting, Directory Table Base (DTB) identification, and obtaining kernel objects. Although most published research works try to solve anti forensics, they deal only with one element among the three steps. Thus, collapse in any of the three steps using the suggested robust algorithms leads to failure in the memory analysis. In this paper, we evaluate the latest memory forensic tools against anti-forensics. Then, we suggest a novel robust algorithm that guarantees the bootstrapping analysis steps. It uses only one kernel data structure called KiInitialPCR, which is a kernel global variable based on the kernel processor control region (KPCR) structure and has many fields with tolerance to mutation. We characterize the robust fields of the KPCR structure to use them for OS fingerprinting, DTB identification, and obtaining kernel objects. Then, we implement the KiInitialPCR-based analysis system. Therefore, we can analyze the compromised memory in spite of the interference of anti-forensics.

Introduction

Memory forensics means the process of acquiring physical memory, which contains volatile data, and collecting evidence from the acquired memory image. Since the Digital Forensic Research Conference memory challenge (Dfrws, 2005) opened in 2005, the extraction of volatile data such as disk encryption key (Mrdovic and Huseinovic, 2011) and user input information (Olajide et al., 2012), as well as the process list (Betz, 2005), has been researched. Memory forensics has become important in digital forensics in that it can extract these volatile data, which is impossible from a hard disk.

Anti-forensic techniques have appeared to prevent memory analysis, due to its importance. According to the definition of anti-forensics (Harris, 2006), anti-forensics to memory analysis is divided into two parts; preventing memory acquisition and memory analysis. In these aspects, research on anti-memory acquisition has progressed (Stüttgen and Cohen, 2013).

Anti-forensic techniques to prevent memory analysis modify fragile signatures on a live system to block the path to evidence (Takahiro Haruyama, 2012). In addition, they can compromise the kernel data structure field, which has a semantic value, to make the memory analysis tools mislead the fields (Prakash et al.). Further, they can construct fake kernel objects to increase the analysis time (Jake Williams, 2014). Among these anti-forensic techniques, the one-byte abort factor showed extreme limitations in modern memory analysis algorithms by modifying only one-byte value used in memory analysis.

To deal with anti-forensic techniques for memory forensics, OS-Sommelier (Gu et al., 2012) and OS-Sommelier+ (Gu et al., 2015) suggest a code-based

* Corresponding author.
E-mail addresses: koungholee@gmail.com (K. Lee), hhu@nsr.re.kr (H. Hwang), kibom@nsr.re.kr (K. Kim), bbong@jnu.ac.kr (B. Noh).

http://dx.doi.org/10.1016/j.diin.2016.04.009

signature generation, which is very effective for OS fingerprinting on a system that is ensured integrity of the kernel code. Additionally, image-based signature (Roussev et al., 2014) is valuable for OS fingerprinting with a corpus of known kernel binaries. Furthermore, profile indexing (Cohen, 2015) identifies kernel versions using only the globally unique identifier (GUID) without known kernel binaries. However, the advance preparation of this method before comparing the profile indexes is long and is weak against interference. The Directory Table Base (DTB) identification of OS-Sommelier is excellent for the ×86 system but is complicated. Research about robust signatures for kernel data structures (Dolan-Gavitt et al., 2009; Lin et al.; Lin) needs to know the OS versions for accurate scanning. In the viewpoint of comprehensive analysis, these research works against anti-forensics handle only one element among the bootstrapping steps for memory analysis — OS fingerprinting, DTB identification, and obtaining kernel objects, which means that any collapse in the robust algorithms will lead to failure in the memory analysis.

Therefore, in this paper, we show the anti-forensic techniques and the limitations of modern analysis algorithms. Then, we suggest a novel memory analysis algorithm based on KiInitialPCR, which assures the bootstrapping analysis. It locates the KPCR structure, which has many robust fields, and uses the relationship among fields that point to global variables for OS fingerprinting. It also acquires DTB in the KPCR structure. In addition, it calculates the relative offset based on KiInitialPCR (instead of the kernel base) to list the kernel objects. Thus, we prove that it is an effective and comprehensive analysis algorithm that uses only one robust structure against anti-forensic techniques.

Section 2 explains how anti-forensics disturb memory analysis, and shows limitation of bootstrapping analysis used by existing tools. It also shows the limitations of current research against anti-forensic techniques. Section 3 evaluates the modern memory analysis methods, including the bootstrapping analysis against anti-forensics. Section 4 introduces the KiInitialPCR-based analysis to deal with anti-forensic techniques. This analysis carves the KiInitialPCR, identifies the OS version, and extracts the process list and the module list using non-modifiable fields in the memory. Section 5 shows the implementation of the suggested analysis procedure. Section 6 discusses the limitations of our system. The final section offers conclusions and directions for future research.

Background

Anti-forensics

Anti-forensics concentrates on how to make investigators fail to collect volatile evidence by modifying critical values used in memory analysis. An attacker with high privilege can modify kernel memory.

The one-byte abort factor attack (Takahiro Haruyama, 2012) shows that important signatures are easily overwritten by malware in such a way that memory analysis can fail to find it. In addition, semantic value manipulation (SVM) attack mutates semantic values, which are data values with important semantic meanings (Prakash et al.). Further, the attention-deficit-disorder (ADD) technique creates fake objects to lead investigators down a wrong path and increases the analysis time (Jake Williams, 2014).

Whereas the SVM and ADD can analyze physical memory images and extract volatile data irrespective if the data are genuine, modified, or fake, attacking bootstrapping analysis like the abort factor makes the analysis fail, which means that the investigator cannot collect any evidence from the physical memory image. Therefore, the attacking bootstrapping analysis is an important problem to be solved first.

Physical memory analysis must perform bootstrapping analysis, which is composed of OS fingerprinting meaning identifying the OS version, acquiring DTB, and obtaining the kernel data structures. Correct OS fingerprinting enables precise parsing of the kernel data structures with accurate structure layout. It also enables precise selection of the analysis algorithms, which are different in different versions. Acquisition of DTB enables reconstruction of the virtual address space, which is the mapping between the virtual and physical addresses. Obtaining the kernel data structures enables us to collect kernel data such as process and thread information.

A potential target of the attacking bootstrapping analysis is all modifiable memory, which does not cause noticeable differences such as crashes in system state with modification, used in the analysis.

Volatility (The Volatility Foundation, 2015), which is a famous memory forensic tool, uses KDDEBUGGER_DATA64 structure, which is known as KDBG, to identify the OS version with *Size* field and get the global kernel variables with fields named same as each variables like PsActiveProcessHead. Then, it uses the EPROCESS structure of the idle process to obtain the DTB with *DirectoryTableBase*.

Memoryze uses the EPROCESS structure of the system process to identify the OS version by matching DISPATCHER_HEADER signatures of all OS (e.g., \×03 \×00 \×1B \×00 and \×30 \×00 \×26 \×00) and by confirming whether the *ImageFileName* field is a “System” string or not. In addition, it obtains the DTB from the system process.

As mentioned in the abort factor, these well-known analysis algorithms use fragile signatures. The attacker can modify the “KDBG” string, “Idle” string or “System” string, which is critical memory values to the analysis, to abnormal value. As we show in Section 3, this technique still can disrupt memory analysis.

Rekall (The Rekall Team, 2015a) gathers the program database (PDB) information from GUID in the RSDS region of the kernel executables and identifies the OS version from the PDB information, which contains the structure layout information and global debugging symbols. It carves the RSDS region with “RSDS” signature and known PDB file names of the kernel executables (e.g., ntoskrnl.pdb). However, the region of the GUID and PDB filename is a modifiable memory.

Further, rekall uses the profile indexing method (Cohen, 2015) known as nt index to deal with the abort factor. In profile generation phase, rekall chooses arbitrarily 10–12 addresses among the virtual addresses of NOP (0×90) instructions preceding the function and of the string literals (e.g. “FILE_VERSION”). These addresses contain debugging