available at www.sciencedirect.com

ScienceDirect

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

Digital Investigation

The persistence of memory: Forensic identification and extraction of cryptographic keys

Carsten Maartmann-Moe[a,*,1], *Steffen E. Thorkildsen*[b], *André Årnes*[c,2]

[a]*Department of Telematics, Norwegian University of Science and Technology, O.S. Bragstads Plass 2B, N-7491 Trondheim, Norway*
[b]*National Criminal Investigation Service, Norway*
[c]*Norwegian Information Security Laboratory, Gjøvik University College, PO Box 191, N-2802 Gjøvik, Norway*

Keywords:
Digital forensics
Data hiding and recovery
Memory analysis
Memory dumping
Applied cryptography
Live analysis
Cryptographic evidence
Incident response
Tool testing and development

ABSTRACT

The increasing popularity of cryptography poses a great challenge in the field of digital forensics. Digital evidence protected by strong encryption may be impossible to decrypt without the correct key. We propose novel methods for cryptographic key identification and present a new proof of concept tool named *Interrogate* that searches through volatile memory and recovers cryptographic keys used by the ciphers AES, Serpent and Twofish. By using the tool in a virtual digital crime scene, we simulate and examine the different states of systems where well known and popular cryptosystems are installed. Our experiments show that the chances of uncovering cryptographic keys are high when the digital crime scene are in certain well-defined states. Finally, we argue that the consequence of this and other recent results regarding memory acquisition require that the current practices of digital forensics should be guided towards a more forensically sound way of handling live analysis in a digital crime scene.

1. Introduction

Cryptography has grown to become one of the most important contributors to privacy and data security in an increasingly interconnected world. The use of cryptography also represents a challenge for digital forensics investigators, as it may be used to hide data that may shed light on the chain of events that constitutes an incident or crime. Since the nature of cryptography makes it attractive for hiding incriminating data, encrypted material encountered often contain exactly the evidence sought by investigators.

In this paper, we aim to study new methods for the identification and extraction of cryptographic keys from the volatile memory of computing devices as part of the *digital forensics process*. In this context, the keys and any encrypted contents may be considered to be *digital evidence* (i.e., digital data that contains reliable information that supports or refutes a hypothesis about an incident (Carrier and Spafford, 2004)) that is part of a *digital crime scene*. Note also that the main property of cryptographic keys in the context of digital forensics is that they may be a necessary prerequisite for the successful decryption of encrypted digital evidence.

Digital investigators are often forced to attempt brute-force and dictionary attacks to gain access to encrypted digital evidence, but these methods cannot circumvent strong cryptography and strong passwords. A paradox is that

* *Corresponding author*.
E-mail addresses: carsten@carmaa.com (C. Maartmann-Moe), steffen.thorkildsen@politiet.no (S.E. Thorkildsen), andre.arnes@hig.no (S.E. André Årnes).
[1] Carsten Maartmann-Moe is currently a Consultant at Ernst & Young in Norway.
[2] André Årnes is currently an Adjunct Associate Professor at Gjøvik University College and a Security and Identity Management Architect at Oracle Norway.
1742-2876/$ – see front matter © 2009 Digital Forensic Research Workshop. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2009.06.002

cryptographic keys may be present in computer memory at the time of the evidence acquisition. However, memory is not always acquired, and there are no standard tools for memory analysis and key extraction based on memory dumps.

The main contributions of this paper is the novel approach to Serpent and Twofish key structure identification and analysis, a method for virtual memory reconstruction, as well as the proposed introduction of cryptographic key searches in memory as part of the digital forensics process. Our results are validated through the implementation of a proof of concept tool and a series of experiments covering three cryptographic algorithms and ten software tools in a virtualized testbed.

The paper is structured as follows. Section 2 contains an overview of related research, Section 3 describes techniques for identifying keys in memory, and Section 4 discusses how to use Windows memory structure to optimize searches. Our experiments and results are presented in Sections 5 and 6, and the implications for the field of digital forensics is discussed in Section 7. Finally, future work and conclusions are provided in Section 8.

2. Related work

The acquisition and analysis of volatile memory for forensics purposes is a relatively immature procedure, even though the concept has been known for a long time (Crescenzo et al., 1999). The memory acquisition process is especially unstandardized, and there exists a large number of different approaches. A good comparison of the available methods for Microsoft Windows operating systems can be found in the paper *Windows Memory Forensics* (Ruff, 2007). The methods for extracting volatile memory ranges from DMA access via FireWire (Dornseif, 2005; Martin, 2007) to simply copying of memory from /dev/mem on Unix-flavor platforms.

Research on the age of freed user process data in physical memory has shown that large segments of pages are unlikely to survive more than 5 min, even on a lightly loaded system (Solomona et al., 2007). However, smaller segments and single pages may be found up to 2 h after initial memory commit. These results may limit the timeframe for successful recovery of cryptographic keys that are left in memory. To counter these issues, Chow et al. have proposed several methods for secure deallocation of sensitive data from memory (Chow et al., 2005). It is nevertheless clear that these results do not mitigate the fact that cryptographic keys need to be present in memory during encryption when using standard computer hardware.

The first approach on cryptographic key search and identification were proposed by Shamir and van Someren in 1998, suggesting the prospect of attacks against mainframes in their article *Playing Hide and Seek with Stored Keys* (Shamir and van Someren, 1998). They propose to use simple statistical and visual methods to locate memory regions that are likely to contain encryption keys. In a more recent article, Pettersson discusses searches for structural properties of the code that is holding the key, by analyzing and ''guesstimating'' the values of surrounding variables (Pettersson, 2007). Ptacek (2008) outlines how to extract and verify RSA keys from memory, using a simple mathematical analysis of the parameters found. On identifying RSA keys, Klein suggests searching for ASN standard prefixes of the DER-encoding, both identifying certificates and private keys in memory (Klein, 2006).

The authors of Volatility describe a hypothetical attack against TrueCrypt (Foundation, 2008), by studying its internal structures and behavior (Walters and Nick, 2007). They do, however, not describe how to locate the different structures in memory, and neither do they discuss the fact that some of these may be paged out, thereby breaking the chain of data structures that leads to the master key if only the memory dump is available for analysis.

Halderman et al. presented a recent breakthrough in their paper *Lest We Remember: Cold Boot Attacks on Encryption Keys* (Halderman et al., 2008). They demonstrate that it is possible to leverage remanence effects[3] in DRAM modules to coldboot the target computer, load a custom OS that extracts the memory to an external drive, locate the key material and finally decrypt the hard drives automatically. We owe the idea to utilize key schedules as a means for identification of cryptographic keys to this paper, and lately considerable effort has been directed at creating usable software for decryption of closed-source systems like BitLocker (Kaplan, 2007; Kornblum, 2008).

Most of these methods treat the memory as a large blob of bytes, although in fact memory is quite structured. Some of the methods suggest skipping duplicate regions and reserved address space, but do not consider to reduce the ''haystack'' by only looking at the probable regions of the memory. In other fields of memory analysis, analysts have dumped the memory address space of a specific process by fetching pages from RAM and swap space. The dumps are sometimes sufficient to verify[4] and even completely reconstruct executable files (Kornblum, 2006). According to several articles (for example, see Schuster, 2006 and Carvey, 2007), these techniques are able to identify trojans, rootkits and viruses that are stealthy and/or armored in Windows memory dumps.

Despite all these contemporary studies, there exist little empirical research on whether cryptographic keys are present in memory at the time of acquisition. In this paper we will demonstrate how to utilize several *search strategies* in combination with cryptographic knowledge to extract key material from volatile memory. We perform controlled experiments that will indicate the probability of a successful key extraction.

3. Cryptographic key identification

For the average end user, a cryptographic key is an abstract notation, hidden by obfuscation layers consisting of password churning and key hierarchies. In reality, symmetric cryptographic keys are just short sequences of random-looking bytes, often 16–32 bytes long. Even so, recent studies suggest

[3] Remanence effects is the effect that all Dynamic Random Access Memory (DRAM) modules keep their state for a period of time (typically a few seconds) before it needs to be refreshed by the memory controller, first mentioned as a security risk in a articles by Anderson (2001) and Gutmann (2001, 1996). The process of utilizing this effect to extract cryptographic keys is known as the ''coldboot technique''.

[4] By using tools like SSDeep by J. Kornblum.