# Cloud and mobile security assurance

## A memory forensics approach

Mark Vella

Department of Computer Science, University of Malta

Cloud computing nowadays constitutes a substantial portion of new enterprise IT spending [4]. Whether signing up for applications deployed as a service, or outsourcing hosting concerns to third parties, cloud computing is taking us back to the days of the data center. Virtualization is a key enabler technology that enables consolidation of hardware utilization and provides the notion of elastic computing whereby hardware resources are allocated on demand [8]. Mobile devices represent a natural fit to this change in the technology landscape, be it a means to compensate for their limited storage or a way to fully justify investment in cloud computing, thereby fueling the bring-your-own-device (BYOD) concept [1].

This new scenario is uncharted territory for security assurance, which attempts to provide a measure of trustworthiness of security-critical objects provided by third parties. In this case, enterprise information is being entrusted to cloud providers and to personal employee mobile devices. When assuring IT infrastructure, code scanning and penetration testing have become established audit practices. However, the sufficiency or even relevance for the cloud/mobility computing infrastructure has to be questioned. Recent global security incidents, namely *Heartbleed* [5] and *Shellshock* [10], show that vulnerabilities can go undetected for years despite existing practices. The case of targeted attacks gets even more complicated [13]. Relevance is also of concern. In the case of cloud computing: should enterprise simply trust the cloud platform's security stack? If the answer is negative, how should a company arrange for a penetrating testing exercise on the cloud-hosted, possibly multi-tenant, infrastructure? Mobile devices complicate the situation even further. One factor that sets them apart from the rest is their hardware constraints. Therefore, should the lack of installed scanning software fail security audits or should these devices be exempt? This decision seems to present one of those impossible practicality-security trade-offs.

The argument being put forward in the proposed research direction is that when the big bug or the resourceful attacker strikes, graceful recovery is key. The intuition is that information retrievable from kernel/user memory should trigger recovery. Periodic or trigger-driven physical memory dumps from virtual machines and mobile devices can be used to assess the current security state across an IT infrastructure. The value proposition is that memory dumps are expected to be comparatively much smaller than disk or network dumps, and can provide a practical solution. On the other hand, the potential effectiveness is still left for exploration. An effective solution should provide a shift from a situation where suspicion of a security breach cannot be verified to one where not only verification is made possible but that also allows for a timely recovery to minimize impact.

The use of memory forensics is not new for the domain of computer security. Malware forensics is a case in point [7], however the predominantly manual and non-structured approach poses the main limitation for assurance. Forensic guidelines suggest for example that process structures in kernel memory could help spot backdoor installations, while investigating kernel data structures such as system service descriptor tables could disclose the presence of rootkits. Security assurance tools require that the whole process be fully automated in order to enable a practical audit process. For this to happen existing guidelines require evolving into proper assurance methods having rigorous measures of effectiveness that can be tied to security risk levels. In the longer term more substantial challenges have to be addressed. Mobile devices may not be always accessible for on-demand memory dumps while their hardware constraints could hinder complete acquisition. Furthermore, the memory forensics-based assurance process should go beyond merely detecting the presence of malware, but rather the entire 'who/when/where/how' forensic spectrum needs answering. This way all damage can be identified, infection vectors closed down and any attack sources identified so that trust in the system can be fully regained. Some of the challenges involved require searching for exploitation residue that could be present in memory. Also, instructions from code sections combined with register and data sections content could enable back-tracing to prior program states of forensic interest. Finally, complete situation rectification could require a generic exploit mitigation mechanism to be put in place rather than a specific patch. Any (memory) forensically derived information could potentially serve as basis for such mitigation.

Existing research efforts provide promising starting points. Leverage of virtual machine technology to provide cloud security assurance has been investigated in terms of tamper-proof virtual machine snapshots [12]. There are also various examples that demonstrate how virtualization facilitates the implementation of various security mechanisms, e.g. control flow integrity (CFI) [2]. Work in automated dynamic analysis of malware also makes use of memory forensic techniques [14]. The problem of automated crash dump analysis also starts off with a memory dump and tackles the problem of program state back-tracing [6], whilst object-code analysis techniques also make use of runtime data structures e.g. call stacks [3]. Finally, memory forensics should complement disk-based forensics [9] and network traffic analysis [11]. Correlating the different forensic levels should be part of the overall research direction in order to maximize the available tool-set for investigators. With proper adaptation and combination with the aforementioned malware forensic guidelines, these existing techniques could pave the way towards a security assurance process that fits today's enterprise computing needs.

## References

1. Bridges, V.: BYOD skyrockets in popularity for 2013  here are the stats to prove it (Dec 2013), http://vbridges.com/2013/12/20/byod-skyrockets-popularity-2013-stats-prove/
2. Carbone, M., Conover, M., Montague, B., Lee, W.: Secure and robust monitoring of virtual machines through guest-assisted introspection. In: Research in Attacks, Intrusions, and Defenses. pp. 22–41. Springer (2012)
3. Chen, X., Slowinska, A., Bos, H.: Who allocated my memory? detecting custom memory allocators in c binaries. In: Reverse Engineering (WCRE), 2013 20th Working Conference on. pp. 22–31. IEEE (2013)
4. Gartner: Gartner says cloud computing will become the bulk of new it spend by 2016 (Oct 2013), http://www.gartner.com/newsroom/id/2613015
5. Heartbleed: The Heartbleed bug (Apr 2014), http://http://http://heartbleed.com/
6. Jin, W., Orso, A.: Bugredux: reproducing field failures for in-house debugging. In: Proceedings of the 2012 International Conference on Software Engineering. pp. 474–484. IEEE Press (2012)
7. Ligh, M., Case, A., Levy, J., Walters, A.: The Art of Memory Forensics. Wiley Publishing (2014)
8. Openstack: Cloud software (Oct 2014), http://http://www.openstack.org/
9. Pal, A., Memon, N.: The evolution of file carving. vol. 26, pp. 59–71. IEEE (2009)
10. Rapid7: Bashbug (shellshock): What is it? how to remediate? (Sep 2014), http://www.rapid7.com/resources/bashbug.jsp
11. Raymond, J.F.: Traffic analysis: Protocols, attacks, design issues, and open problems. In: Designing Privacy Enhancing Technologies. pp. 10–29. Springer (2001)
12. Srivastava, A., Raj, H., Giffin, J., England, P.: Trusted VM snapshots in untrusted cloud infrastructures. In: Research in Attacks, Intrusions, and Defenses. pp. 1–21. Springer (2012)
13. Symantec: 2014 Internet security threat report, volume 19 (Apr 2014), http://www.symantec.com/security_response/publications/threatreport.jsp
14. Yin, H., Poosankam, P., Hanna, S., Song, D.: Hookscout: Proactive binary-centric hook detection. In: Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 1–20. Springer (2010)