AFR: Automatic Multi-Stage Forensic Data Retrieval

David Gugelmann
ETH Zurich, Switzerland

Dominik Schatzmann
ETH Zurich, Switzerland

Vincent Lenders
armasuisse, Switzerland

ABSTRACT

The investigation of malware infections in enterprise networks is today a tedious task with a lot of manual intervention in order to find the scattered relevant bits and bytes from infected hosts. We propose in this work AFR, a framework for automatic multi-stage forensic data retrieval, that automatically analyzes and retrieves network, memory and disk data to preserve the evidence of host compromise at a central location. AFR performs automated malware analysis using traditional intrusion detection techniques like network intrusion detection systems and anti-virus software but combines the resulting alarms in real-time to proactively retrieve and archive data that is relevant for retrospective investigations. The proactive retrieval approach reduces the manual work load of IT administrators while significantly improving the likelihood that volatile data is collected before it vanishes. We show that proactive storing of selected memory and disk dumps is feasible and scales over time in virtualized thin client enterprise environments.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

Forensic investigations, network security, network monitoring, automatic data retrieval

1. INTRODUCTION

IT administrators use network intrusion detection systems (NIDS) to protect corporate networks from malware. However, to investigate, analyze and mitigate possible malware infections, the network-related information provided by NIDS is often not conclusive. Therefore, more information, such as the content of the RAM and the disk of the suspected malware infected PC must be collected. Especially RAM data should be gathered as fast as possible, otherwise data might get overwritten before collection. In addition, the inherent high false positive rate of NIDS makes manual collection of evidence cumbersome.

As a solution to these problems, we propose in this work AFR, an automatic multi-stage forensic data retrieval framework. AFR uses a chain of data collectors and analyzers for network, memory, and disk data. Based on the output of sensors located earlier in the chain, a decision is made whether to continue data collection. The IT administrator is not immediately involved but only when sufficient evidence for an infection has been collected. If sensors later in the chain show with high confidence that a false positive occurred, data retrieved earlier is partially deleted. Additionally, we employ an algorithm to interlace different snapshots to store consecutive snapshots of the same host efficiently.

AFR has several advantages: (1) the probability of successfully preserving volatile data is higher since network, memory and disk data are immediately retrieved on alerts, (2) the work load on the administrator is reduced because data does not have to be collected manually, (3) the number of false positives is decreased and (4) host-based data becomes as easily accessible for the IT administrator as network-based data.

The effectiveness of AFR depends on the underlying intrusion and malware detection sensors. Forensic data is only collected for detected infections, therefore zero-day exploits are out of scope of this framework.

2. AFR FRAMEWORK

Overview. The architecture of AFR consists of data retrieval, malware analysis and storage components. Each component is applied in a chain to network, memory and disk data of host PCs in order to decide which evidence should be automatically retrieved and stored at a central location. An example is shown in Fig. 1; first the network is monitored (1). In case of suspicious activity, full network traffic is recorded for the involved hosts and a RAM snapshot is automatically retrieved from these hosts (2). If RAM analyzers indicate malware activity, an alarm is raised to the IT administrator and a disk image (3b) is automatically retrieved. Otherwise the recording of network traffic is stopped and the retrieved RAM image is deleted (3a). The captured network traffic is still archived to cope with false negatives of RAM analysis. Next, the disk analyzers scan the disk image, if sufficient evidence for an infection is found, the incident's priority is increased and the IT administrator is alerted again (4b). Otherwise the disk image is deleted, the previously recorded RAM image could additionally also

CoNEXT Student'12, December 10, 2012, Nice, France.

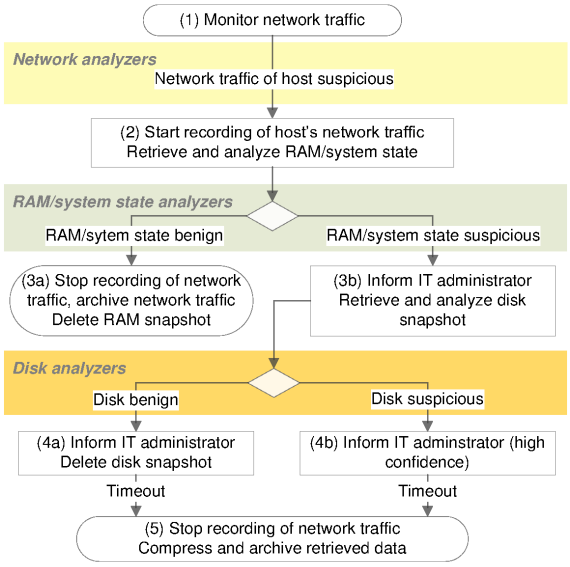

Figure 1: Example analysis and collection chain.

be deleted, but we decided against this because of memory-only malware and to cope with false negatives in disk analysis. In any case, the recording of full network traffic stops after a timeout and the retrieved data is compressed and archived (5). The architecture of AFR is independent from the methods and tools used for data retrieval and analysis. We discuss possible methods and tools in the following.

Data retrieval. With `tcpdump` and related tools, there are many solutions available for recording network traffic. Collecting trustworthy information from physical hosts is more challenging because malware on the host can tamper with the data collection process. This problem does not arise in thin client environments, which is the focus of our work. In thin client environments, RAM and disk images can be generated by the hypervisor and thin client instances are usually derived from a base disk image, to which only the differences are stored for every instance. This difference file can be used directly as disk snapshot.

Sensors. As shown in Fig. 1, our approach relies on different kind of analyzers to assess the suspiciousness of the retrieved data. For *network traffic* analysis, NIDS such as `Snort` and `Bro`, but also behavior based analysis such as [2] are possible solutions. *RAM snapshots* can be analyzed using anti-virus (AV) software, but also correlation analysis between information from a RAM snapshot and information reported by the host could be used to identify stealth malware. Solutions for RAM image analysis are, e.g., the `Volatility Framework` and [4, 1]. Analyzing *disk snapshots* is the traditional domain of AV software.

Storage. A major concern with respect to proactive forensic data collection is the required storage space. Using a NIDS to filter network traffic has been presented in [3]. We show how consecutive memory and disk snapshots can be stored efficiently for the case where hosts stay infected over long time. We found that a linear increase of total snapshot volume can easily be avoided by transposing snapshots before compressing and storing them. Transposing means that snapshots are split into small blocks (currently 1024 Byte) and then all first blocks of consecutive snapshots are concatenated to a stream followed by all second blocks, and so on. Interlacing consecutive snapshots moves similar content close to each other such that stream compressors, `gzip` in this case, can compress the stream much better.

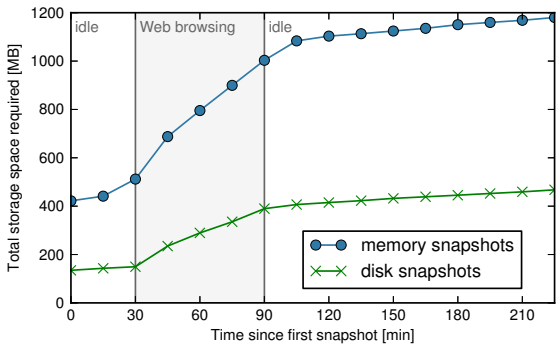

Figure 2: Total volume of consecutive snapshots.

We measured the space required for storage of consecutive disk and RAM snapshots of a thin client system with 1 GiB of RAM, a base disk image of 19.9 GB and a difference file of 418 MB at the time of the first snapshot. A snapshot has been created every 15 min. Web browsing was done for 1 h starting half an hour after the first snapshot, otherwise the host has been idle. Fig. 2 shows the storage space required to store all snapshots when using gzip with interlacing. Only 1180 MB of storage are required for 16 memory snapshots. Using gzip without interlacing would require 6361 MB. 468 MB are required to store 16 disk snapshots, the disk snapshots account for 12025 MB without compression and for 4638 MB when using gzip without interlacing.

3. CONCLUSIONS AND FUTURE WORK

We have presented a framework for proactive capturing of forensic data and have shown that storing consecutive host snapshots scales over time in thin client environments. As future work, we plan to evaluate the performance of different analysis and collection chains.

4. ACKNOWLEDGMENTS

This work was partially supported by the Zurich Information Security Center. It represents the views of the authors.

5. REFERENCES

[1] A. Case, L. Marziale, and G. G. Richard. Dynamic recreation of kernel data structures for live forensics. *Digital Investigation*, 7, Supplement:32–40, 2010.

[2] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, 35(4):229–240, Aug. 2005.

[3] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider. Enriching network security analysis with time travel. *SIGCOMM Comput. Commun. Rev.*, 38(4):183–194, Aug. 2008.

[4] A. Schuster. Searching for processes and threads in microsoft windows memory dumps. *Digital Investigation*, 3, Supplement:10–16, 2006.