Draft of a Dynamic Malware Detection System on Trustworthy Endpoints

Andreas Speier · Christofer Fein · David Bothe
Eric Reich · Norbert Pohlmann

Institute for Internet Security
Westphalian University of Applied Sciences Gelsenkirchen
{speier | fein | bothe | reich | pohlmann}@internet-sicherheit.de

Abstract

Malware infected computer systems can be found with increasing evidence in private and commercial fields of use. Always exposed to the risk of a "Lying End-Point", an already manipulated security application that pretends to run on a clean computer system, the demand for new security solutions continues to rise. Project iTES ("innovative Trustworthy Endpoint Security"), government-funded by the German Federal Ministry of Education and Research, introduces a new system to enhance security while preserving usability. Based on an existing virtualized system which diversifies the software to a specific form of use, the project aims to develop new sensors to monitor the system dynamically and deliver real-time responses.

1 Introduction

Malware infected computer systems can be found with increasing evidence in private and commercial fields of use. Even technically advanced countries fall victim to severe damages caused by malicious software. Prevention of such attacks has to result in hardening computer systems against malware activity. Critical applications like online banking and e-commerce introduce a profitable field for criminal individuals, which significantly raises the need for trusted data processing even more. Conventional security solutions available today already offer thorough countermeasures, but are often exposed to manipulation by malware themselves. Always being subject to the risk of a "Lying End-Point", an already manipulated security application that pretends to run on a clean computer system [SSDD07], the demand for new security solutions continues to rise.

Project iTES[1] introduces a new system to enhance security while preserving usability. The system architecture needs to guarantee the reliability of a security solution with provable integrity to prevent attacks. Typical computer systems will be assembled regarding private used and professional used software components to serve as a reference for behavior analysis (see section 2.4).

Common Trusted Computing technologies combined with virtualization create a secure software environment to isolate malware, which provides a basis for innovative and trustworthy security systems. A core intention of iTES is to combine already existing security technologies and

1 http://ites-project.org

advance them through innovative developments. Continuous usage of security software will be fortified by virtualization and integrity-measures of its components. Another important part is the development of software sensors (see section 2.5) for dynamic malware behavior analysis and detection by a security software guest system.

In section 2.3 the architecture of a physical client is outlined. The inner structure, like the main security concept (see section 2.1-2.2) and the software sensors (see section 2.5) are described in detail. The software separation is mentioned in section 2.3.

The secure environment with the multiple client concept and the central component are described in general in section 3.

2 System Architecture

Common trusted computing technologies offer manipulation detection and possibilities to attest configurations of computer systems. Virtualization adds the separation of single applications within strong isolated compartments. The hypervisor is the interface between the host system and a virtualized guest system. In our system this interface is also intended to integrate sensors to perform dynamic malware analysis (see section 2.4).

2.1 Integrity

Proving the integrity is one of the main goals in securing a system. The integrity of the guest system has to be examined before the startup routine by calculating checksums of the virtual machine configuration files. A complete check of the virtual machine image is a time consuming task and may take several minutes as shown in table 1. In the shown example a single 13 GB OS image has been subjected to the hash algorithms shown, on a middle class business notebook using some well-known hash algorithms. On a system as described in this paper, there will be at least four guest system images. Therefore a smarter integrity check has to be developed to perform rapidly before the virtual machines are permissible to be started. A pre-boot investigation will be done on security relevant components of the virtualized system. These components have to be compared with their pre-defined secure states. A deviation from the defined initial state has to be identified as abnormal configuration [Pohl08]. In this case the virtual machines start procedure has to be denied and the configuration of the system as a whole must be reestablished as described in section 2.2.

Table 1: Mean checksum calculation time (s) on 13GB image file, read from disk

Algorithm / Disk Technology	**HDD**	**SSD**
md5	~112 s	~49 s
sha1	~128 s	~85 s
sha256	~117 s	~51 s
sha512	~120 s	~61 s

2.2 Availability

System security includes integrity and confidentiality, but also to steadily provide defined services [ALRL04]. Abnormal changes to a system can influence the availability of a special service, or the system itself. In this case the service or the whole system must be restored. The work of this project will also concentrate on developing several ways for system remediation. Simply restoring a previous virtual machine snapshot has to be the worst case method in case of a detected malicious anomaly. This leads to the loss of all user data, stored on the specific virtual machine. Developing intelligent methods to restore minor parts of the system is one aim of this project.

2.3 Client Architecture

The architecture needs a separated compartments for different types of tasks. Implementing these environments provides strong isolation that can – in case of an infection - prevent malware from spreading inside the system and affecting security critical components. Malware in one compartment can compromise a single virtual machine, but not the entire system [Micr10].

Therefore one of the first steps during the project was to identify the different software products frequently used on computer systems in home and work environments. We analyzed different usage statistics to compile a comprehensive list of employed software packages and their corresponding versions by the end of September 2012 ([StOwl12], [Stai12], [Webm12]). Due to the virtualized architecture of the client system we had to categorize the software into different compartments. The aims of the different compartments were chosen to achieve the highest possible usability while prioritizing security concerns. The main software categories for the user of the system are:

- Internet related software
- Office related software
- Work related software
- Banking software
- Browser
- Email

The client architecture will include an additional compartment for security and data analysis software as described in section 2.4.

2.3.1 Internet Related Software

Internet related software is software that needs to be connected to the Internet to perform well. This includes instant messengers for communication, P2P software for file sharing or content on demand services to retrieve multimedia content. The connection to the Internet is a high risk for software and data stored inside this compartment. Using sensitive data inside this compartment is not recommended.

2.3.2 Office Related Software

Office related software is software that is used in a secure work environment and can run without a direct Internet connection. This includes office suites [Webm12], document readers, multimedia software and image processing tools. The aim is to separate common malware infected software from the Internet to preserve the security of sensible data, e.g. invoices transmitted as PDF documents.

2.3.3 Work Related Software

Special applications are needed in specific professional fields. Different types of software like CAD tools, accounting or IDE software can be installed in this compartment, which is built to isolate this mainly unknown and not controlled software.

2.3.4 Banking Software

A banking suite is placed in a special compartment. Due to security reasons, no other software is installed alongside. Banking tasks can be performed securely without being influenced by security issues caused by other software.

2.3.5 Browser

A special compartment is erected to support secure Internet browsing. The separation from other software components impedes infection of and from these components. By enforcing this barrier between the different tasks, the security for handling sensitive tasks like e-commerce is provided.

2.3.6 Email

This compartment is made for email handling only. A secure e-mail client provides a barrier for malware distributed by email attachments like keyloggers and scareware. The combination with a policy based data flow management results in a usable environment regarding the handling of attachments, which can be opened in other appropriate compartments.

2.4 Client Security

It is possible to group the different compartments into security level based domains, providing different levels of connectivity, e.g. to the network as illustrated in figure 1.

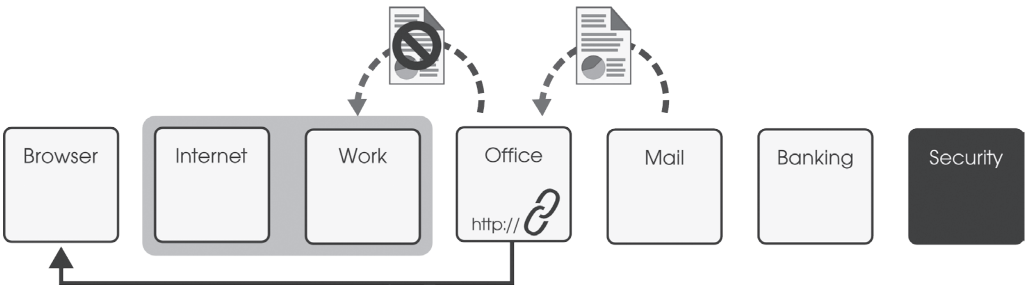

Fig. 1: Compartment Overview

The host internal communication between these domains, e.g. for copy and paste functions, are managed by a predefined set of patterns. For usability reasons, the functionality of clicking a link in an offline compartment or the ability to send files as mail attachments is provided through a policy based data flow management. This will verify the source of the data and redirect it based on its internal rules to the browsing or email compartment. Consider a situation where an employee receives a purchase order by email containing a HTTPS-link to a prepared shopping cart. Using this web-link, the system will enforce a forwarding to a secure browsing compartment by some policy. This avoids risks like the interception of credit card information possible in an insecure environment.

Another special type of compartment is the "High Security Compartment" providing an isolated environment for security related software such as firewall, anti-virus and sensor data processing software. Communication with this compartment is only allowed in one direction for sensor data (see section 5). The security compartment is not usable for the standard user, but for the system manager. This inhibits accidental changes and targeted attacks of malware against security software [MSMP10].

A conservative anti-virus tool matches the signatures of a specific file with its database to identify and classify malware. These signatures are created in an automated way by security software vendors on known malware samples. The Problem is, that obfuscated malware can change the binary code, with the result that the signature changes [ESKK08]. If this sample was not found by security software before, the malware can perform the intended malicious actions on a system. The approach of analyzing malware without its execution is called static analysis.

Static analysis can be performed on different representations like source code or binary code. Due to the restricted availability of malware source code, static analysis on binary code leads to multiple problems e.g. in context of self-modifying malware and runtime dependent execution values [ESKK08].

The second approach is to analyze the runtime behavior of malware within a running environment. This is called dynamic malware analysis. Dynamic analysis is a way to observe the action of executed malware on different ways. This includes e. g. function call monitoring, function parameter analysis, information flow tracking and instruction tracing [ESKK08].

Our aim is to transfer dynamic malware analysis into a trustworthy system of endpoints in combination with conservative static analysis techniques to get a real time response system. Intensive data analysis and gathering data with the assistance of software sensors are not resource-efficient tasks for a system with virtual guests. One professed goal is to preserve usability, while enhance security by behavior analysis.

2.5 Software Sensors

In general, software sensors in this project can be divided into two categories: in-box and out-box sensors.

In-box sensors are placed inside the virtual machine. The gathered data from this location is mainly generated in the user space and sent to the analysis software residing inside the "High Security Compartment".

The disadvantage of internal sensors is the danger of them being manipulated by malware into generating a benign data stream. This problem is known as the "Lying End-Point" [SSDD07]. To face this issue the project also uses out-box sensors, which, assuming a correct working hypervisor, cannot be manipulated by running malware inside the virtual machine. The aim is to produce a corresponding out-box sensor for each data flow an in-box sensor produces. This complement is used to validate the internal information flow. Manipulation of the in-box sensors should be recognized by this approach.

The developed sensors will be evaluated for their usability in productive systems by the following concerns: In-box sensors are also in danger of being detected by malware scanning the system for security software and may incite a different behavior. The development of out-box sensors for productive systems is very important, because they cannot be detected by malware searching for security software. These sensors will use a hypervisor interface for monitoring vital parts of the guest system by using virtual machine introspection [GaRo03]. The disadvantage of out-box sensors is that they are more cost intensive in terms of system resources.

3 Overall Architecture

The overall architecture is modeled with a company like environment in mind, employing several physical client systems with a central management entity. This architecture provides some advantages over a stand-alone solution: Changes in the compartment arrangement can be managed easily from a central instance, enabling convenient administration and a consistent security policy throughout the entire system. The centralized management of the compartment configuration improves security by preventing misconfiguration of individual client systems.

Observing a single client only yields a limited view of its behavior, making it difficult to identify malicious activities solely relying on the analysis performed inside its own "High Security Compartment". To improve the classification of behavior the gathered data should optionally be sent to a central entity called the "Central Component". By aggregating the data generated by multiple client systems comparison and anomaly detection methods can be employed, allowing a vastly improved a broader view on the global system state.

The communication between client nodes and the central entity should be performed using two different modes of operation. A verbose mode will be used to transmit every collected data item, yielding a vast amount of input data e.g. for machine learning algorithms.

A less communication intense approach will only transmit behavioral records once an anomaly has already been detected by the client sided "High Security Compartment". The data then is subjected to further analysis by the "Central Component".

Due to the client being able to run in not trustworthy environments, the entire communication traffic will be protected using TLS, providing the confidentiality of the data and the authenticity of the client's identity. For the purposes of authenticity, digital certificates will be used, which provide the verifiability of the client's identity.

4 Conclusion

Behavior sensitive malware analysis on physical clients is a major step forward in security system development. Our system architecture will combine this dynamic approach with a detection technique based on the experience of multiple clients, trusted computing approaches and an arrangement of virtual guests in on one physical client.

First detection will be done with analysis software in the “High Security Compartments” on each physical client. A system wide entity, the “Central Component”, will gather all information about the state of each physical client. Optionally an auxiliary analysis will be made on this high performance computer system. Every anomalous incident will be identified and classified to gather information for all clients of the system. The use of virtual machines for different software classes will prevent the spreading of malware. Trusted computing technologies like assurance of virtual image integrity will help to attest a secure security state of the whole system.

The main theoretical design work for this framework is done and the implementation is in progress. A usable prototype with underlying distributed system will be available soon.

The present overview of the project benefits from the input of student assistants and our project partners. These are: Chair of System Security (Ruhr University Bochum), Sirrix AG security technologies and Avira Operations GmbH & Co. KG. We also thank the German Federal Ministry of Education and Research for funding.

References

[ALRL04] Avizienis, Algirdas and Laprie, Jean-Claude. and Randell, Brian and Landwehr, Carl: Basic Concepts and Taxonomy of Dependable and Secure. In IEEE Transactions on Dependable and Secure Computing Vol. 1, No. 1. 2004, S. 11-33.

[ESKK08] Egele, Manuel and Scholte, Theodoor and Kirda, Engin and Kruegel, Christopher: A survey on automated dynamic malware-analysis techniques and tools. In: ACM Computing Surveys, vol. 44. s.l. : ACM New York, 2008.

[GaRo03] Garfinkel, Tal and Rosenblum, Mendel. A virtual machine introspection based architecture for intrusion detection. In: Proc. Network and Distributet Systems Security Symposium. 2003.

[Micr10] Microsoft. Intel TXT Homepage. [Online] 29. 04 2010.

[MSMP10] Microsoft. Microsoft Malware Protection Center – Encyclopedia TrojanDownloader:Win32/Perka.A. [Online] 29. 04 2010.

[Pohl08] Pohlmann, Norbert. Trusted computing. Ein Weg zu neuen IT-Sicherheitsarchitekturen. s.l. : Vieweg, 2008.

[SSDD07] Sahita, Ravi and Savagaonkar, Uday R. and Dewan, Prashant and Durham, David: Mitigating the Lying-Endpoint Problem in Virtualized Network Access Frameworks. In: Managing Virtualization of Networks and Services. Berlin, Heidelberg : Springer, 2007, S. 135-146.

[Stai12] Statista.com. Statista Messenger Statistics. [Online] 10. 10. 2012.

[StOw12] StatOwl.com. StatOwl Browser Statistics. [Online] 12. 10 2012.

[Webm12] WebmasterPro.com. WebmasterPro Office Suits Statistics. [Online] 10. 10. 2012.