Comprehensive Testing of Software and Hardware Data Security Tools Using Virtualization

A. V. Epishkina, A. M. Kanner and T. M. Kanner

Abstract The article considers the need to use modern virtualization tools in the process of developing, assembling and testing software and hardware data security tools. Such data security tools include hardware components that implement key security functions, but also impede the use of number of testing tools. The use of virtualization tools for software and hardware data security tools allows to perform security checks, which are difficult to test on physical computing hardware, for example: security functions that run independently of the operating system of computing facility or before starting the user session. However, using virtualization tools may degrade the "purity" of the testing process, so it is necessary to ensure that software implementation of the hardware devices' virtual connection interfaces and virtual machine components comply with existing standards and specifications. Depending on the connection interface of hardware components of data security tools, it is possible to use the built-in capabilities of virtualization tools or existing AMD IOMMU and Intel Vt-d technologies to redirect these components to a virtual environment.

Keywords Testing of software and hardware data security tools · Virtualization tools · Testing programs for data security tools for protection against unauthorized access

A. V. Epishkina · A. M. Kanner (✉) · T. M. Kanner
National Research Nuclear University MEPhI, Kashirskoe Highway, 31, 115409 Moscow, Russia
e-mail: kanner@mail.ru

A. V. Epishkina
e-mail: ann-arsky@yandex.ru

T. M. Kanner
e-mail: sheikot@mail.ru

S. Y. Misyurin et al. (eds.), *Advanced Technologies in Robotics and Intelligent Systems*, Mechanisms and Machine Science 80, https://doi.org/10.1007/978-3-030-33491-8_9

1 Introduction

Modern virtualization tools greatly simplify the process of developing, assembling and testing software and data security tools (DST), for example, allowing to do as follows [1]:

- To emulate the most popular hardware platforms—x86, arm, ppc64, s390x, etc.;
- To emulate a large number of software platforms—various versions and bitness of operating systems (OS) or other application software;
- To conduct load testing, i.e. to emulate a large number of users, processes, connection sessions, etc.;
- To perform testing of functions that run independently of the OS, before the OS is booted, or before the user session is started;
- To perform sequential testing on several computers—virtual machines (VM), with succession of some hardware resources.

At the same time, an important difference between virtualization tools and, for example, containerization tools during development and testing lies in the ability to fully emulate hardware platforms, taking into account all their basic features, which is absolutely necessary when developing system software and DST.

As compared to software DST, software-and-hardware DST include hardware components made as an independent or connected physical device (for example, with a USB or PCI interface). Such hardware components can perform key security functions on the one hand, and prevent the use of certain tools that simplify the process of development, assembly and testing, including virtualization tools, on the other hand.

The main factors that do not allow to fully use certain virtualization tools for the development, assembly and testing of software-and-hardware DST are as follows:

1. Inability to redirect all types of hardware components into a virtual environment;
2. Possibility of errors in the implementation of the virtualization tools themselves (for example, in the tools used to redirect the required hardware devices), which may be absent when using physical computers;
3. Dependence of some security functions on the characteristics of physical computers, which cannot be emulated using virtualization tools;
4. Inconsistency of some virtualized entities (processor, BIOS/UEFI, interrupts, etc.) with existing standards and specifications [2].

Thus, real physical computers shall be ideally used for software-and-hardware DST, which complicates the development, assembly and testing processes, but allows to sufficiently ensure accuracy of these processes [2]. However, virtualization tools greatly simplify scalability, and also allow to perform a number of tests that cannot be automated on physical computers. In this regard, it is necessary to find out which modern virtualization tools (for example, VMware, Oracle and Parallels, as well as those built into the operating system—Hyper-V, KVM, etc.) may be used in relation to software-and-hardware DST, and under what conditions.

2 Materials and Methods

When testing software-and-hardware DST, one shall at first take into account that both software and hardware components can be in different states. In this regard, to be able to compute [3] security function tests, it is necessary, first, to ensure reachability of all these states, and, second, to implement all possible functions of transition from one state to another. Therefore, upon completion of each test, if necessary, either the DST is transferred to another state required for the next test, or the DST is returned to the state in which this security tool was before the start of the current test (initial/reference state). In terms of state reachability when using virtualization tools, it is necessary to first consider the possibility of redirecting hardware components of the DST to the VM.

When using virtualization tools, the least difficulties arise in relation to software-and-hardware DST, which have a mobile hardware component that implements security functions that interact with the OS environment of the computer [1]. As a rule, virtualization tools support redirection of such devices into the VM, and there are corresponding drivers to support virtual connection interfaces in the OS. Difficulties may arise in the process of using virtualization tools for testing software-and-hardware DST, whose security functions do not interact with the OS and are implemented as a part of the computer, for example, those related to the possibility of initiating boot using the DST or taking control before the VM OS is booted. However, there are no such difficulties in those virtualization tools that use software implementations of BIOS (and other components of the computer), which are similar to the hardware ones and correspond to the specifications, but not interact directly with the hardware of the physical computer. For example, KVM (Kernel-based Virtual Machine) uses SeaBIOS that conforms to the specification [4, 5].

When using virtualization tools to test security functions implemented on the basis of a stationary hardware component, it is necessary to redirect it to the virtualization environment using the following technologies—AMD I/O Virtualization Technology (AMD IOMMU) or Intel Virtualization Technology for Directed I/O (Intel VT-d) [6, 7]. The basis of these technologies lies in the use of a special input/output memory management unit (IOMMU), which allows various peripheral devices to be directly used in the VM through interruption mapping and direct memory access (DMA) tables. The tools implementing security functions based on a stationary hardware component include, for example, controllers with PCI/PCI-express interface. For AMD IOMMU or Intel VT-d technologies to work correctly, they shall be supported by a processor, a motherboard, a system/internal computer software (BIOS or UEFI), and a computer OS, where the virtualization tool is used. In the absence of such technologies in one of the above computer components, it will be impossible to redirect the hardware component implementing the security functions into the VM.

Thus, when testing security functions of the software-and-hardware DST implemented on the basis of a stationary hardware component (including those with the PCI connection interface) using virtualization tools, it is necessary to use a specialized computer, in which all components support AMD IOMMU or Intel VT-d technologies. To enable redirection of such devices into the VM, it is also necessary to ensure support for such technologies in the OS. For example, in case of Linux and KVM virtualization environment, in order to redirect devices to the VM using Intel VT-d, the OS kernel with the following configuration parameters [1] shall be used:

- CONFIG_PCI_STUB—to provide the possibility to "detach" devices from OS drivers and redirect them to the VM environment;
- CONFIG_INTEL_IOMMU—to enable support for Intel VT-d using DMA mapping tables (to translate all physical memory calls from the VM);
- CONFIG_IRQ_REMAP—to support the mapping of interruptions.

In the future, to redirect devices to the KVM virtualization environment, before booting the VM one shall first detach the device from the OS driver, then attach the device to the driver that will be used to redirect it to the virtualization environment (pci_stub):

```
export BASE_ADDRESS="0000"
export PRODUCT_ID="03:00.0"
export VENDOR_ID="1795 0700"
modprobe pci_stub
echo "${VENDOR_ID}" > /sys/bus/pci/drivers/pci-stub/new_id
echo "${BASE_ADDRESS}:${PRODUCT_ID}" \
      > /sys/bus/pci/devices/${BASE_ADDRESS}:${PRODUCT_ID}/driver/unbind
echo "${BASE_ADDRESS}:${PRODUCT_ID}"> /sys/bus/pci/drivers/pci-stub/bind
```

After that, the VM may be booted using the device location on the bus as a parameter, for example: "*-device pci-assign,host = 03:00.0,id = amdz0*", where 03:00.0 is the interface to which the PCI device is connected. The structure of complete xml-description file for the virtual machine in libvirt with automatic redirection of some device is presented below:

```
<domain type='kvm' id='1'>
<name>VM_NAME</name>
<description>VM_DESCRIPTION</description>
<memory unit='KiB'>2000000</memory>
<os>
   <type arch='VM_ARCH' machine='pc-1.3'>hvm</type>
</os>
<devices>
   <emulator>/usr/bin/qemu-system-VM_ARCH</emulator>
   ...
   <hostdev mode='subsystem' type='pci' managed='no'>
      <source>
         <address domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
      </source>
   </hostdev>
   ...
   </devices>
</domain>
```

where VM_NAME is the virtual machine name, VM_DESCRIPTION—it's description and VM_ARCH—architecture used (i386, x86_64 or other).

Redirection of mobile hardware components can be performed through the use of well-known Vendor ID and Product ID numbers (VID and PID) or of their location on the bus corresponding to the used interface (let us denoted it as INTERFACE). In the libvirt environment, automatic device redirection can be specified in the xml-description file:

```
<hostdev mode='subsystem' type='INTERFACE1' managed='no'>
        <source>
        <vendor id='0xa420'/>
        <product id='0x5426'/>
        </source>
</hostdev>
<hostdev mode='subsystem' type='INTERFACE1' managed='no'>
        <source>
        <address bus='001' device='003'/>
        </source>
</hostdev>
<hostdev mode='subsystem' type='INTERFACE2' managed='no'>
        <source>
        <address domain='0x0000' bus='0x03' slot='0x00' function='0x0'/>
        </source>
</hostdev>
```

The process of redirection of such type of devices could also be done with the help of built-in commands of libvirt:

- ‘virsh attach-device VM_NAME dst-hardware-interface-passthrough.xml’—to attach redirected device specified in some xml-description file to the virtual machine VM_NAME;
- ‘virsh detach-device VM_NAME dst-hardware-interface-passthrough.xml’—to detach redirected device specified in some xml-description file from the virtual machine VM_NAME.

In terms of performing various functions of software-and-hardware DST transition from one state to another [3], it should be borne in mind that tests shall be performed first with a permanently connected hardware component, and then reconnect it during operation and upon completion of each test using embedded capabilities of virtualization tools or using auxiliary hardware [3, 8]. This is due to the fact that when a real user uses some software-and-hardware DST, the extraction and connection of the hardware component may occur at an arbitrary point in time, and may also be required due to the specific features of the hardware component itself.

In a number of cases, software-and-hardware DST testing programs shall consist of several modules, for example:

- When testing one security function on several computers (VM), the testing program shall include modules that operate in each VM, as well as a module in the computer OS environment with a virtualization environment that performs management, sequential switching of the DST hardware components between the VMs and consolidation of the test results.
- When verifying the DST security functions implemented prior to the launch of the user session (for example, identification and authentication) or before the boot of the OS, testing should be performed in the computer OS with the virtualization environment installed, while the modules of the testing program should also run in the VM or DST OS (depending on the security function). This is due to the fact that it is not always possible to determine the result of such security functions in the computer OS, in which the VM is running. The testing program can work only with native objects of the computer OS and cannot intercept and analyze the results of its actions in the VM OS, including using technologies such as VMware Unity mode for VMware virtualization tools [9], Seamless mode in VirtualBox [10], or Coherence for Parallels virtualization tools [11].

Based on the above, it is possible to suggest an algorithm for testing security functions of the software-and-hardware DST using virtualization tools. This algorithm is applicable to all software-and-hardware DST, except for those whose security functions are implemented independently of the computer. For such DST virtualization tools can be used only when interacting over a network in order to emulate a large number of users, processes and connection sessions. The proposed algorithm allows to make the tests of security functions computable, which are not computable on physical computers. The above algorithm consists is as follows:

1. It is necessary to use virtualization tools and their standard capabilities to redirect the hardware component of the DST implementing the security functions to the VM. In this case, the software implementation of the security tool connection

interface, as well as of the VM components, which are similar to the components of physical computer, shall comply with existing standards and specifications (for example, libvirt-controlled KVM).
2. To test security functions implemented on the basis of a stationary hardware component, it is necessary to use specialized computers, in which all components (processor, motherboard, BIOS or UEFI, OS) support AMD IOMMU or Intel VT-d technologies, and the virtualization tool being used shall allow redirection of this type of components.
3. To perform reconnection of hardware components using standard features of virtualization tools or auxiliary hardware to implement various functions of software-and-hardware DST transitions from one state to another.
4. To implement the modules of testing programs in both the computer OS and the VM OS (DST) when testing security functions that are implemented before the launch of the user session or before the boot of the OS, and use several computers (VM) with sequential connection of the hardware component, if necessary.

3 Results and Discussion

The described results allowed us to develop a software complex "Testing of security functions of software-and-hardware data security tools", which allows to fully automate the processes for assembling, testing and verifying [12, 13] various types of software-and-hardware DST. Among other things, this made it possible to organize regression testing, in which any changes in the original DST source codes are automatically checked for violations of the security functions, which previously functioned correctly.

In particular, let us consider the characteristic features of using the developed testing programs for Accord-X access control subsystem, which is built on the basis of Accord-TSHM trusted start-up hardware module [14]. Some of the security functions of Accord-X are implemented independently of the computer OS (trusted start-up), and the rest are implemented on the computer OS (identification and authentication, access control, integrity control, creation of an isolated software environment). At the same time, in order to carry out identification and authentication, it is required to submit hardware identifiers that do not directly implement security functions, but are necessary for their implementation (computability of their tests). The programs testing security function of Accord-X carry out tests in two stages: first, before the boot of the VM OS with respect to the security functions that run independently of the computer OS, and then directly in the VM OS with respect to the security functions that are implemented in the computer OS, in both cases taking into account the possibility of connecting various additional hardware components (identifiers). Thus, using the developed software, it became possible to automatically perform the following sequence of actions:

1. Assemble the components of the software-and-hardware DST for various supported software and hardware platforms.
2. Install and configure access control software depending on the VM OS, in which the testing is carried out: installation of packages, configuration of OS components and bootloader, basic configuration of the DST.
3. Restart the VM and test operability of the DST with the basic settings (correctness of the DST activation, identification and authentication when logging in the OS with hardware identifiers, etc.).
4. Generate file system objects and access rules for testing DST security functions with various parameters (combination of access attributes, recursion rules, access and confidentiality levels) and in various conditions (various methods of getting access or changing integrity).
5. Run security function tests (about 100,000 access requests with different settings and combinations of security functions) and fix possible inconsistencies.
6. Send test results to the program "Verifier of software-and-hardware DST" [12] for further automatic verification.

The described sequence allows to check all the security functions of Accord-X with various combinations and in all possible states of the software and hardware components of the DST, in which such functions can be computable, as well as to justify the attainability and maintenance of an absolutely isolated software environment for users to ensure impossibility of violating the applicable security policies [15]. It should be borne in mind that when using virtualization tools, the testing accuracy may deteriorate when another level is introduced—the virtual environment hypervisor, in which errors can arise that are not related to functioning of the DST itself or computer hardware components. Hardware platforms emulated using virtualization tools can be found not to correspond to similar physical computers. In such cases, operability of the DST in the virtual environment does not guarantee its operability in a real computer with similar characteristics. However, similar guarantees cannot be often provided even when using physical computers produced by different manufacturers [2].

References

1. Kanner, T.M.: Features of using virtualization for testing software and hardware data security tools (In Russian). Inf. Secur. **18**(3), 416–419 (2015)
2. Sinyakin, S.A.: Characteristic features of Accord-TSHM compatibility in modern computers (In Russian). In: XVIII International Conference, Complex Data Protection 2013. Elektronika Info, vol. 6, pp. 142–144. Brest, Belarus (2013)
3. Kanner, T.M.: Applicability of software testing methods to software and hardware data security tools. Glob. J. Pure Appl. Math. **12**(1), 167–190 (2016)
4. Wong, A.: Breaking Through the BIOS Barrier: The Definitive BIOS Optimization Guide for PCs. Prentice Hall PTR, Upper Saddle, New Jersey (2004)
5. Zimmer, V., Lewis, T., Rothman, M.: Harnessing the UEFI Shell: Moving the Platform Beyond Dos. Intel Press, Germany, Berlin (2010)

6. AMD I/O Virtualization Technology (IOMMU) Specification Revision 1.26. http://developer.amd.com/wordpress/media/2012/10/34434-IOMMU-Rev_1.26_2-11-09.pdf. Last accessed 27 June 2019
7. Intel Virtualization Technology for Directed I/O (VT-d) Architecture Specification. http://www.intel.com/content/www/us/en/embedded/technology/virtualization/vt-directed-io-spec.html. Last accessed 27 June 2019
8. Kanner, T.M.: The effectiveness of using supporting tools for testing software and hardware DST (In Russian). Inf. Secur. Quest. **2**, 9–13 (2017)
9. VMware Workstation Documentation Center. Use Unity Mode. https://pubs.vmware.com/workstation-9/index.jsp. Last accessed 27 June 2019
10. Oracle VM VirtualBox User Manual. Seamless windows. https://www.virtualbox.org/manual/ch04.html#seamlesswindows. Last accessed 27 June 2019
11. Parallels Desktop User's Guide. Working in Coherence. http://download.parallels.com/desktop/v4/docs/en/Parallels_Desktop_Users_Guide/23413.htm. Last accessed 27 June 2019
12. Kanner, A.M., Kanner, T.M.: Applying a mathematical approach to interpreting the results of testing software and hardware data security tools during the verification process. J. Eng. Appl. Sci. **14**(10), 3482–3491 (2019)
13. Kanner, T.M.: Adaptation of existing verification methods for software and hardware DST (In Russian). Inf. Secur. Quest. **1**, 13–19 (2018)
14. Kanner, A.M., Ukhlinov, L.M.: Access control in GNU/Linux (in Russian). Inf. Secur. Quest. **3**, 35–38 (2012)
15. Kanner, A.M.: Correctness of data security tools for protection against unauthorized access and their interaction in GNU/Linux. Glob. J. Pure Appl. Math. **12**(3), 2479–2501 (2016)