A Performance Analysis of ARM Virtual Machines Secured Using SELinux

Michele Paolino[✉], Mian M. Hamayun, and Daniel Raho

Virtual Open Systems, Grenoble, France
{m.paolino,m.hamayun,s.raho}@virtualopensystems.com
http://www.virtualopensystems.com

Abstract. Virtualization of the ARM architecture is becoming increasingly popular in several domains. Thus security is one of the main concerns in modern virtualized embedded platforms. An effective way to enhance the security of these platforms is through a combination of virtualization and Mandatory Access Control (MAC) security policies. The aim of this paper is to discuss the performance overhead of MAC-secured virtual machines. We compare the I/O performance of a KVM/ARM guest running on a SELinux host with the one of a non-secured VM. The result of the comparison is unexpected, since the performance of the SELinux based VM is better than the non-secured VM. We present a detailed analysis based on a modified version of SELinux running on an ARM core, and highlight the main causes of the observed performance improvement.

Keywords: ARM virtualization · SELinux · KVM ARM · VM security · MAC virtual machines · Mandatory access control (MAC)

1 Introduction

The ARM architecture is expanding from embedded systems to server, automotive and High Performance Computing (HPC) platforms. The use of virtualization is rapidly increasing in these platforms to save power through consolidation, to isolate applications and to deploy multiple operating system instances on shared hardware resources. As the use of virtualization technology becomes common place in enterprise and end-user markets *e.g.* Data Centers, NFV (Network Functions Virtualization) systems, Android devices, CPS (Cyber Physical Systems) *etc.*, new security aspects have emerged, such as protecting virtual machines from potential host based attacks.

A hypervisor or Virtual Machine Monitor (VMM) creates virtual instances of the CPUs, memory and interrupts to provide an illusion of a real machine in software. When the VMM implements full virtualization, it provides hardware isolation for these resources exploiting hardware features *e.g.* Virtualization Extensions, IOMMU and GIC in ARM platforms. Other resources such as

F. Cleary and M. Felici (Eds.): CSP Forum 2014, CCIS 470, pp. 28–36, 2014.
DOI: 10.1007/978-3-319-12574-9_3

device peripherals (network, disks *etc.*) or shared memory are isolated in software by the hypervisor and virtualized through emulation or para-virtualization. This constitutes the concept of isolation using virtualization, which is valid from the guest point of view but not from the host perspective. Especially for a privileged user in a standard Discretionary Access Control (DAC) environment, VM's resources are accessible without any restrictions. This means that a cloud administrator may read the disk data and sniff network traffic of its customers' virtual machines. Moreover, an attacker can compromise the host system (even from a virtual environment) and perform un-authorized operations over the resources that belong to the VMs.

Security issues that specifically affect virtual environments have been classified as: communication between VMs or between VMs and host, VM escape, VM monitoring from the host, VM monitoring from another VM, denial of service, guest-to-guest attacks, external modification of a VM and external modification of the hypervisor [14]. Most of these security threats, aim to compromise isolation between guests or between guest and host *e.g.* using the CPU cache [23] or directly assigned devices [13] to gain privileges or access un-authorized data. To mitigate these threats, hypervisors are provided with strong access control mechanisms [22] like the Mandatory Access Control (MAC) and Role Based Access Control (RBAC) [12]. In fact, in every virtualized system such as a cloud, the primary challenges for data security are the separation of sensitive data and access control mechanisms [20].

This paper presents a performance overview of the KVM/ARM VMs that leverage MAC policies to secure virtual resources. The Linux kernel provides different alternative implementations of MAC security policy such as SELinux, TOMOYO, AppArmor and SMACK. None of these is clearly better than the others but SELinux is considered the most mature and widely deployed amongst Linux enhanced security mechanisms [17]. The KVM hypervisor has been selected for this evaluation as it exploits the standard SELinux implementation. In fact, the most important alternative VMM for ARM *i.e.* XEN, has its own MAC implementation wrapped in Xen Security Modules (XSM) [2].

We compare the performance of two VMs: one running on a host using the DAC security policy and the other executed in SELinux environment. This comparison shows unexpected results, as the performance of SELinux based VM is better than the non-secure VM. We isolate and discuss the key factors behind this behavior using a modified version of SELinux. To the best of our knowledge, this is the first SELinux based performance analysis for KVM/ARM virtual machines.

This paper is organized as follows: Sect. 2 describes SELinux and the Linux Security Modules (LSM). Section 3 gives details on the hardware and software platform used to gather test results that are presented in Sect. 4. Related work is described in Sect. 5 and potential future directions are given in Sect. 6.

2 Security in the Linux Kernel

By default the Linux kernel uses DAC security policy, which is based on users and groups. This policy is easy to use but has significant drawbacks. In fact, DAC allows the owner of a resource to freely delegate rights over it. Moreover, only two types of roles are supported: super user and normal users. The former (also known as root), may be a security threat as it has complete control of the system. This is particularly undesirable in multi-user, multi-tenancy systems such as cloud, server, NFV and CPS environments.

To overcome the security issues of DAC, Linux combines it with the MAC security policy, where a system-wide mechanism controls access to objects *e.g.* a socket, a disk file *etc.*, and an individual subject *e.g.* a process, a VM *etc.*, cannot alter it [6].

2.1 Linux Security Modules and SELinux

To avoid the proliferation of security solutions that perform invasive modifications to the Linux kernel, support for security solutions has been provided through an abstraction layer known as the Linux Security Modules (LSM). It enables the implementation of MAC policies as loadable kernel modules avoiding the necessity to deal with long and difficult to maintain patches. LSM allows modules to mediate access to kernel objects by placing hooks in the kernel code just ahead of access to them [25]. These hooks are scattered through-out the kernel and have been classified as task, program loading, file-system, IPC, module and network hooks [24]. A security module implements some or all of these hooks.

In 2001 SELinux was initially presented to the open source community as a kernel patch by the National Security Agency(NSA), and was later re-implemented as LSM module [19]. It is an implementation of the Flask OS security architecture [5] and its MAC policy is based on Type Enforcement (TE) that can also provide Role Based Access Control (RBAC). The Flask OS's main capability is to separate security access control decisions from their enforcement [1]. This feature has been inherited in SELinux, where the Security Server takes the security access control decisions and the LSM hooks enforce them [7]. Furthermore, SELinux has

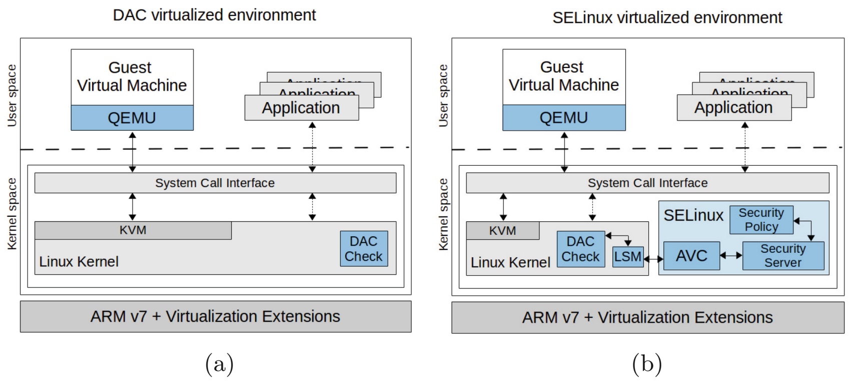

Fig. 1. DAC and SELinux based virtualization environments

a third component known as Access Vector Cache (AVC), which is designed to speed-up the access validation decisions. The AVC maintains a cache of decisions made by the Security Server for subsequent accesses [7]. Figure 1 shows the DAC and SELinux based virtualization environments.

2.2 Disabling the SELinux AVC

When comparing a guest in a DAC virtualization environment (Fig. 1a) with a VM in a SELinux host (Fig. 1b), the later shows better I/O performance. This result is unexpected given that SELinux introduces at-least two different sources of overhead: the first one comes from the LSM layer and the second is due to the access control decision making infrastructure *i.e.* SELinux performs a security check every time a subject wants to execute an operation on an object. So this additional cost *should* lower the performance of SELinux host VMs. In order to explain these results and evaluate its performance impact on the overall security system, we *disable* the main component designed to improve the SELinux performance *i.e.* the AVC.

We modify the `avc_has_perm_noaudit()` function, which performs permissions checks in every access. The permission check is firstly delegated to the AVC cache and if the result is not found (an AVC miss), the request is forwarded to the Security Server. In order to oblige the Linux kernel to always go through the Security Server, we force a cache miss for each request. To include the lookup time in our measurements, we force the cache miss *after* the AVC lookup function. This modification aims to keep the SELinux source code changes as simple as possible. In fact, a complete removal of the AVC would result in important modifications to the existing code as it has been included in SELinux from very early stages, and it is fully integrated into it. In addition, this enables us to measure the AVC cache miss influence on an implementation that is very similar to the mainline SELinux.

3 Hardware/Software Platform and Benchmarks

The Texas Instruments OMAP5-uEVM board has been used to perform these tests. It is equipped with two ARM Cortex-A15 MPCore (1.5 GHz), 2 GB of DDR3L RAM and a 16 GB MicroSD card (Class 6). Although Ubuntu 12.04 is the most widely used distribution on ARM, it does not officially support SELinux so we installed Fedora 20 as a host on the OMAP platform. To create and manage virtual machines we used QEMU 1.7.91 and libvirt 1.2.2 [10]. The mainline kernel v3.14.0 is used for the host; for the DAC virtualization environment (Fig. 1a) it is compiled without any security support (*i.e.* no LSM) and for the MAC secured virtualization environment (Fig. 1b), it is compiled with SELinux support and booted in targeted enforcing mode.

The VM runs Ubuntu 12.04 (kernel v3.12.0-rc7) with 256 MB of RAM and is pinned to a physical processor. The virtio para-virtual drivers have been used for both network and disks. The VM's disk image is stored on the host local storage. The *noop* I/O scheduler and EXT4 file-system have been used in both guest and host systems.

The *iozone* and *netperf* software benchmarks are used for the guest file-system and network tests, respectively. In fact, I/O is the most important reason for interactions between the guest and host systems. The disk tests have been performed with different file sizes *i.e.* 4 KB, 100 KB, 1 MB, 2 MB and 10 MB. The smallest file size is equal to the block size of EXT4 file-system, while the higher values are small-to-medium sizes that are commonly found in different use-cases. To prevent any caching mechanism between the VM and host, we disable caches in the *virtio* and *iozone* configurations. The performance evaluation of SELinux within the virtual machines is out-of-scope of this paper.

4 Performance Evaluation and Results

In this section we present some experimental results on I/O performance of the ARM virtual machines. All of the disk performance figures show the average results of 13 file-system operations for 5 different file sizes (65 in total), and each test has been repeated 30 times. In Figs. 2 and 3, a negative result means that the VM on SELinux host is faster as compared to the DAC host VM. Figure 2 presents a comparison between two guests: the first running in a DAC environment and the second on a SELinux host. In this case, the SELinux host based VM is faster in 38 out of 65 tests (58 % negative results).

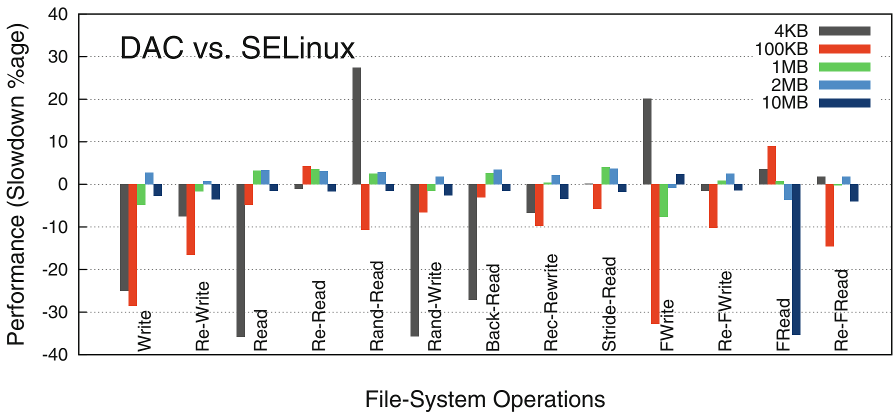

Fig. 2. VM disk performance with a standard SELinux host (with AVC)

These results are unexpected for the reasons discussed in Sect. 2.2. So we neutralize the SELinux AVC cache and obtain the results shown in Fig. 3. These results highlight the overall impact of AVC cache on the disk performance. It is interesting to see that 8 out of 65 results are still negative (12 %), where 7 results are greater than −8 % of slowdown percentage and the most negative one

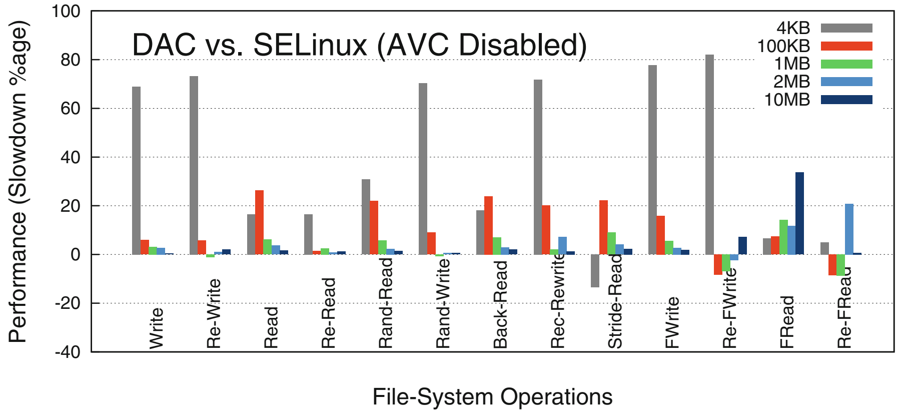

Fig. 3. VM disk performance with a SELinux host (AVC disabled)

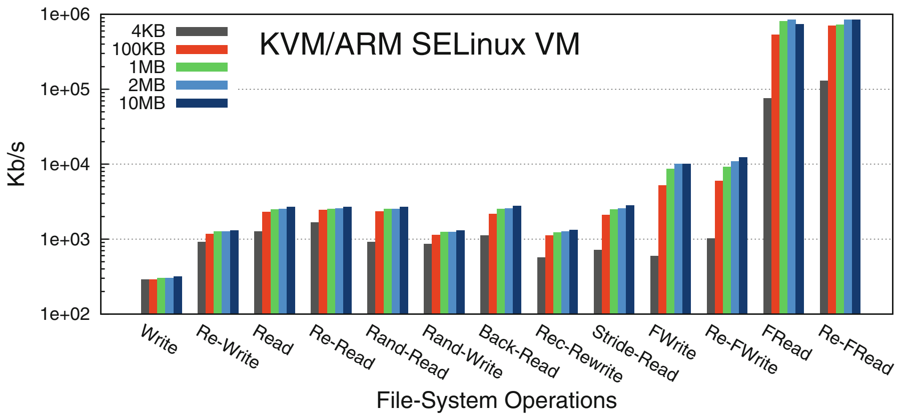

Fig. 4. VM absolute disk performance on SELinux host (with AVC)

Table 1. VM streaming I/O performance results (*netperf*)

	SELinux (with AVC)	SELinux (AVC disabled)	DAC host
TCP_STREAM	45.96 Mbps	14.23 Mbps	41.79 Mbps
UDP_STREAM	114.57 Mbps	23.99 Mbps	81.75 Mbps

Table 2. VM request/response I/O performance results (*netperf*)

	SELinux (with AVC)	SELinux (AVC disabled)	DAC host
TCP_RR	630.37 Tps	362.50 Tps	615.07 Tps
UDP_RR	653.93 Tps	377.01 Tps	641.18 Tps

is about −13 %. We consider these results as mostly an experimental anomaly and in part due to the LSM framework. In fact, there are examples in literature where LSM performs better than DAC [25]. Finally, Fig. 4 shows the absolute disk performance of a KVM/ARM guest on a SELinux host.

For the network benchmarks, a similar approach has been taken. We compare the performance of three VMs: the first on a DAC host, the second running on SELinux with the AVC disabled, and the third running on a full SELinux host (leveraging the AVC). Two tests have been performed for both TCP and UDP protocols: bulk data transfers (TCP_STREAM and UDP_STREAM) and request/response performance (TCP_RR and UDP_RR). These results are presented in Tables 1 and 2, where the bandwidth and packet processing rates are shown in Mega-bits per second (Mbps) and Transactions per second (Tps), respectively. In both cases, similar to the disk benchmark results, we can claim that SELinux VM is faster than the DAC guest. These results also show that AVC has a significant impact over the network performance of the guest VMs.

5 Related Work

Park [11] did a MAC performance analysis of the Android OS using TOMOYO Linux and claims a performance loss of around 25 %. On the same operating system, Shabtai [18] did a SELinux based performance analysis that results with a negligible performance loss. In this study, the authors confirmed two cases of SELinux speed-up, but without any analysis of the possible reasons. In addition, Nakamura [9] measured the performance of SELinux specifically tuned for resource-constrained devices and Wright [24,25] measured the performance overhead of LSM security framework on the x86 architecture, claiming a nearly zero overhead.

Coker and Vogel [3,21] ported SELinux to different ARM platforms while Fiorin [4] developed a hardware accelerated AVC to speed-up performance. None of these works take into account virtualized environments.

Other studies include the Mandatory Access Control implementation directly in the hypervisor (vHype and XEN, Sailer [15,16]) to improve the management and run-time security of the system. Lastly, Nahari [8] proposed a secure embedded Linux architecture by means of virtualization, SELinux and ARM TrustZone.

6 Conclusions and Future Work

We provided a detailed I/O performance analysis of a KVM/ARM guest running on a SELinux host. We compared these results with a guest running without any security enhancements *i.e.* on a DAC host. Our test results show that virtual machines running in a DAC environment are slower than virtual machines running on a SELinux host. We discussed the main causes of this performance improvement, and finally we strongly recommend the use of SELinux in any virtualized environment *e.g.* Data Centers, NFV systems, Android devices, CPS *etc.*, for both security and performance enhancement.

Future work will include an analysis of the LSM framework impact on the guests performance in systems enhanced with MAC security policies. In addition, it will be interesting to study the scalability of KVM/ARM VMs on SELinux hosts, analyzing the performance while increasing number of guests in the system. Finally we will investigate acceleration methods for those systems which cannot exploit MAC security.

Acknowledgment. This research work has been supported by the FP7 TRESCCA project under the grant number 318036.

References

1. Barr, J.: The Flask Security Architecture. Comput. Sci. **574**, 6 (2002)
2. Coker, G.: Xen security modules (XSM). Xen Summit, pp. 1–33 (2006)
3. Coker, R.: Porting NSA security enhanced linux to hand-held devices. In: Proceedings of the Linux Symposium, Ottawa Linux Symposium (2003)
4. Fiorin, L., Ferrante, A., Padarnitsas, K., Regazzoni, F.: Security enhanced linux on embedded systems: a hardware-accelerated implementation. In: 17th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 29–34. IEEE (2012)
5. Lepreau, J., Spencer, R., Smalley, S., Loscocco, P., Hibler, M., Andersen, D.: The flask security architecture: system support for diverse security policies. In: SSYM'99 Proceedings of the 8th conference on USENIX Security Symposium (2006)
6. Lindqvist, H.: Mandatory access control. Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901 87 (2006)
7. Mayer, F., Caplan, D., MacMillan, K.: SELinux by Example: Using Security Enhanced Linux. Pearson Education, Prentice Hall (2006)
8. Nahari, H.: Trusted secure embedded linux. In: Proceedings of 2007 Linux Symposium, pp. 79–85. Citeseer, Ottawa Ontario, Canada:[sn] (2007)
9. Nakamura, Y., Sameshima, Y.: SELinux for consumer electronics devices. In: Proceedings of Linux Symposium, pp. 125–133 (2008)
10. Paolino, M.: sVirt Security for KVM Virtualization on OMAP5 uEVM. http://www.virtualopensystems.com/en/solutions/guides/kvm-svirt-omap5/
11. Park, J., Kim, B., Kim, S.R., Yoon, J.H., Cho, Y.: Performance analysis of security enforcement on android operating system. In: Proceedings of the 2011 ACM Symposium on Research in Applied Computation, pp. 282–286. ACM (2011)
12. Pék, G., Bencsáth, B., et al.: A survey of security issues in hardware virtualization. ACM Comput. Surv. (CSUR) **45**(3), 40 (2013)
13. Pék, G., Lanzi, A., Srivastava, A., Balzarotti, D., Francillon, A., Neumann, C.: On the feasibility of software attacks on commodity virtual machine monitors via direct device assignment. In: ACM Symposium on Information, Computer and Communications Security (ASIACCS) (2014)
14. Reuben, J.S.: A survey on virtual machine security. Helsinki University of Technology (2007)

15. Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J.L., van Doorn, L.: Building a MAC-based security architecture for the xen open-source hypervisor. In: 21st Annual Computer Security Applications Conference, 10 pp. IEEE (2005)
16. Sailer, R., Valdez, E., Jaeger, T., Perez, R., Van Doorn, L., Griffin, J.L., Berger, S., Sailer, R., Valdez, E., Jaeger, T., et al.: sHype: secure hypervisor approach to trusted virtualized systems. Technical report, RC23511 (2005)
17. Schreuders, Z.C., McGill, T., Payne, C.: Empowering end users to confine their own applications: the results of a usability study comparing SELinux, AppArmor, and FBAC-LSM. ACM Trans. Inf. Syst. Secur. (TISSEC) **14**(2), 19 (2011)
18. Shabtai, A., Fledel, Y., Elovici, Y.: Securing android-powered mobile devices using SELinux. IEEE Secur. Priv. **8**(3), 36–44 (2010)
19. Smalley, S., Vance, C., Salamon, W.: Implementing SELinux as a linux security module. NAI Labs Rep. **1**, 43 (2001)
20. Thapliyal, M., Mandoria, H.L., Garg, N.: Data security analysis in cloud environment: a review. Int. J. Innovations Adv. Comput. Sci. **2**(1), 14–19 (2014)
21. Vogel, B., Steinke, B.: Using SELinux security enforcement in linux-based embedded devices. In: Proceedings of the 1st international Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), p. 15 (2008)
22. Vollmar, W., Harris, T., Long Jr., L., Green, R.: Hypervisor security in cloud computing systems. ACM Comput. Surv., 1–22 (2014)
23. Weiß, M., Heinz, B., Stumpf, F.: A cache timing attack on AES in virtualization environments. In: Keromytis, A.D. (ed.) FC 2012. LNCS, vol. 7397, pp. 314–328. Springer, Heidelberg (2012)
24. Wright, C., Cowan, C., Morris, J., Smalley, S., Kroah-Hartman, G.: Linux security module framework. In: Ottawa Linux Symposium. vol. 8032 (2002)
25. Wright, C., Morris, J., Kroah-Hartman, G., Cowan, C., Smalley, S.: Linux security modules: general security support for the linux kernel. In: Foundations of Intrusion Tolerant Systems (OASIS'03), p. 213. IEEE Computer Society (2003)