Published in IET Information Security
Received on 15th August 2012
Revised on 5th December 2012
Accepted on 10th December 2012
doi: 10.1049/iet-ifs.2012.0252

ISSN 1751-8709

Systemic threats to hypervisor non-control data

Baozeng Ding[1,2], Yeping He[1], Yanjun Wu[1], Jiageng Yu[1,2]

[1]*National Engineering Research Center for Fundamental Software, Institute of Software, Chinese Academy of Sciences, Beijing 100190, People's Republic of China*
[2]*Graduate University, Chinese Academy of Sciences, Beijing 100049, People's Republic of China*
E-mail: sploving1@gmail.com

Abstract: Hypervisors are becoming a widespread virtualisation layer in current computer systems. Recent successful attacks against hypervisors indicate that they face the similar integrity threats as traditional operating systems. Current approaches that secure hypervisors mainly focus on code or control-data integrity, without paying attention to non-control data integrity. In this study the authors construct attacks that target hypervisor non-control data to demonstrate which types of data within the Xen hypervisor are critical to system security. It shows privilege, resource utilisation and security policy related data are vulnerable to return-oriented programming or DMA attacks. By modifying their values from one to another, the whole system's performance will be affected. By discussing current approaches that secure hypervisors, which are not suitable for non-control data, the work is to motivate new innovation in this area to protect them.

1 Introduction

Current attacks usually alter control data of the target program to execute injected malicious code or out-of-context library code, such as return-to-library attacks [1]. Control data refers to data that are loaded to processor program counter at some point in program execution. It mainly includes return address and function pointer. By hijacking control data, attackers can alter the control flow of the target program and execute the code they want. Although control-data attacks are considered the most critical security threats [2, 3], it is pointed that non-control-data attacks can cause the equivalent severity as control-data attacks and many real-world software applications are susceptible to such attacks [4]. The attackers can compromise the target application by altering its non-control data, such as configuration data, user identity data and decision-making data. Besides the application level, there are also attacks against kernel non-control data [5], such as resource wastage attack and entropy pool contamination. Another typical such kind of attack is process hiding [6], which modifies the contents of the kernel linked lists used for process accounting and scheduling to hide a malicious process from system utilities. To the best of our knowledge, there is no research work that pays attention to non-control-data on the virtualisation layer.

Hypervisors are becoming a widespread virtualisation layer in current computer systems. A hypervisor virtualises the real system hardware to provide isolated run-time environment and allow resource sharing. Thus operating systems can run on one physical machine as virtual machines (VMs). Since the hypervisor has a much smaller code base than conventional OSes and thus can be better inspected to reduce software bugs, it is usually assumed to be secure and utilised to improve security of VMs running on it [7–10]. However, current commercial hypervisors are quite complex and still have a large code base. For instance, Xen 4.1.2 contains about 350 K source lines of code [11]. Recent successful attacks against hypervisors [12, 13] demonstrate that they are not as secure as assumed. Considering such attacks, there are several approaches to harden hypervisors [14–16]. They focus on code integrity [14, 15] or control data integrity [16] of hypervisors. To our knowledge, there is no research work to secure hypervisors from the aspect of non-control data. In this paper, we demonstrate which kinds of data within the Xen hypervisor are critical to system security, towards further research work to protect them. By our experiments, it shows three types of non-control data are critical to the whole system security. It includes privilege related data, resource utilisation related data and security policy related data. By manipulating these data, attackers can elevate a VM's privilege or cause VM performance degradation or result in security policies within Xen failure.

2 Attacks

We experimented with several types of non-control data within the Xen hypervisor. It showed the following kinds of data are critical to system security:

- Privilege related data
- Resource utilisation related data
- Security policy related data

In this section, we present attacks that target on these data and show why each is critical to system security.

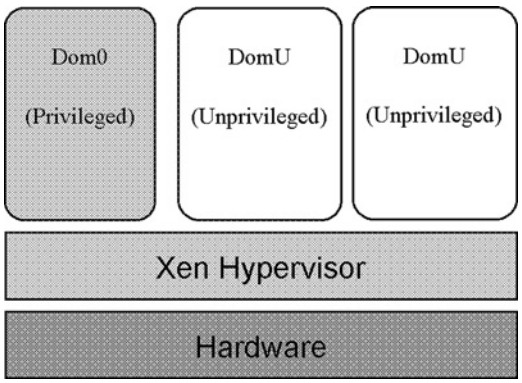

Fig. 1 *Xen architecture*

Dom0 is the only privileged domain, which has the rights to manage all the unprivileged domains (DomUs)

2.1 Privilege related data

2.1.1 Background: The Xen architecture is shown in Fig. 1. Xen runs VMs in environments known as domains. When system boots, Xen is first loaded by the bootloader, then Xen loads its first domain dom0. It is the only administrative domain that is responsible for creating, starting, pausing, unpausing and stopping other domains. In contrast, other domains that dom0 creates have no such privilege so they are called domU, unprivileged domain. If attackers escalate a domU's privilege, this domU then has the same privilege as dom0.

2.1.2 Attack description: The data that controls whether a domain is privileged or not is stored in a domain structure within the hypervisor, shown in Fig. 2. It stores the basic information for a VM, including the following fields: domain_id, the identification of a domain. For dom0, it is 0 and it is increased one by one when creating a new domU. Another field is next_in_list, which is used to link these domain structures together, so by traversing this field the domain structure of a VM can be found identified by its domain_id field. The is_privileged Boolean filed is used to control whether a VM is privileged or not. Generally, only in dom0 this value is set to be 1. In all the domUs, this value is set to be 0. To escalate a domU's privilege, we need to modify this value allocated for that domU from 0 to 1.

```
struct domain {
    domid_t domain_id;
    void *sched_priv;
    struct domain *next_in_list;
    bool_t is_privileged;
    ...
}
```

Fig. 2 *Domain structure*

The domain_id field is the identification of a domain. The next_in_list field links another domain structure. The is_privileged field controls whether a domain is privileged or not

First, it is needed to locate the address of the domain structure of that domU. Then the offset of is_privileged field in that structure can be used to finally get the address to change the value. We use the next_in_list and domain_id fields to get the domain structure of the domU. In Xen, the domains are linked in ascending order by their domain_id. There is a global variable dom0 in Xen that points to dom0's domain structure. Hence we can traverse the next_in_list field starting at dom0 variable. When we get the next domain structure, we first check whether the domain_id equals the id of domU whose privilege to be escalated. If it is not, we continue to use the next_in_list field to get the next address of domain structure. If it is, we locate the address of is_privileged by adding its offset in the structure, which can be easily obtained by analysing the domain structure.

2.1.3 Impact: In our implementation, we run three VMs on the Xen 4.1.2 hypervisor. One is dom0 with domain id zero. The other two is a domU with domain id one (domain 1) and a domU with domain id two (domain 2). We launch a return-oriented programming attack (ROP) [17], which takes advantage of existed hypervisor code instead of modifying any code to change the is_privileged field of domain 1. The implementation details of the attack could be found in our work [18]. After that, domain 1 can pause, unpause or stop domain 2. It can also start other domUs. In cloud environment, an attacker can stop other VMs illegally after escalating his VM's privilege. It may also cause the server denial of service by starting many VMs maliciously.

2.2 Resource utilisation related data

2.2.1 Background: Xen introduces virtual central processor unit (VCPU), to ensure that every running VM receives some CPU time. In the view of VMs, they consider VCPU to be the real resources. In fact, the hypervisor is responsible for mapping VCPUs to physical CPUs. Xen provides several schedulers, the default of which is the credit scheduler. Each VM has two properties associated with it, a weight and a cap. The weigh determines the share of the physical CPU time that VM receives, whereas the cap represents the maximum. Weights are relative to each other. If domain 1 has a weight of 256 and domain 2 has a weight of 128, it has the same effect as domain 1 has a weight of 128 and domain 2 has a weight of 64. In contrast, the cap is an absolute value, representing a proportion of the total CPU that can be used. If it is set to be 100, then the VM can use one physical CPU at most. If set to be 400, it can use four CPUs. When it is set to be 0, it means there is no upper limit. Attackers may decrease a VM's performance by manipulating such resource utilisation related data.

2.2.2 Attack description: The data that represent the weight and cap properties are stored in a structure called csched_dom in Xen, as shown in Fig. 3. For each domain, a field called sched_priv in its domain structure, shown in Fig. 2, is initialised to pointer to this domain's csched_dom structure. Hence by identifying the address of domain structure of a domU, we can finally find the address of resource utilisation related data about it. First, we use the same way described in Section 2.1 to locate a domU's domain structure. Then we use the offset of sched_priv to get address of the domU's csched_dom structure. At last, we use the offset of weight and cap fields to get addresses of this domU's weigh and cap properties.

doi: 10.1049/iet-ifs.2012.0252

```
struct csched_dom {
    struct domain * dom;
    uint16_t   weight;
    uint16_t   cap;
     ...
};
```

Fig. 3 *csched_dom structure*

The weigh field determines the share of the physical CPU time that VM receives and the cap represents the maximum

```
struct chwall_binary_policy {
   u32 max_types;
   domaintype_t *conflict_aggregate_set;
                        /*[max_types] */
     ...
}
struct chwall_binary_policy chwall_bin_pol;
```

Fig. 4 *chwall_bin_pol global variable*

The max_types field is the number of Chinese Wall types. The conflict_aggregate_set field stores all the ChWall-types that are conflicted with any of the ChWall-types of running VMs

2.2.3 Impact: In our experiment, we run three VMs on the Xen hypervisor. One VM is dom0, with four VCPUs allocated. The other two are domain 1 and domain 2, which have exactly the same configure initially. They both have 256 M memory and 4 VCPU resources allocated. The weight of them is the default value 256 and the cap is 0. First, in order to see how the weight property affects a VM's performance, we pin both domains' VCPUs to the same physical CPU and then only alter the weight value of domain 2, from 256 to 1. We separately run UnixBench [19] in both domains at the same time. Unixbench provides a basic reference of the performance of a Unix-like system. The result is higher, its performance is better. We run each test three times. As is shown in Table 1, the front two rows list the average results of domain 1 and domain 2 separately. For the same type of test, we use the result in domain 2 divide the value in domain 1 to compare their performance. It is expressed as percentages in the third row of the table. It demonstrates that the performance of domain 2 is about 70 percentage of domain 1, although the ratio of their weights is 1:256. This is because dom0 is idle and domain 2 can receive CPU time that belongs to dom0. Second, in order to see the importance of the cap property, we alter its value of domain 2 from 0 to 1, which means it can use all the CPUs before but now can only use 0.01 CPU at most. The average results are shown in Table 2. We can see that the performance of domain 2 is nearly one percent of that of domain 1. The experiments show that the weight and cap value are both important to a VM's performance, of which the cap is more crucial. Attackers can modify such kind of data to make a VM degrade its performance greatly or even denial of service.

2.3 Security policy related data

2.3.1 Background: Xen security modules (XSM) [20] are a generalised security framework for Xen to improve security of virtualisation environment. It is derived from Linux security modules [21] and is flexible to add access control modules. Current existing XSM modules mainly include access control module (ACM) and flask. Our target is ACM module. ACM, also known as IBM sHype [22], allows Xen to apply various security policies to VMs. One of them is Chinese Wall policy [23], which is used to ensure that certain VMs with conflicted workload types cannot run on the same system at the same time. It defines a set of Chinese Wall types (ChWall-types) and uses them to define conflict sets. A VM is assigned ChWall-types in function of the workload it can run and VMs with the ChWall-types in the same conflict set cannot run at the

Table 1

Test								
VM	Arithmetic	Dhrystone	Execl	File copy (1024B)	Pipe throughput	Process creation	Shell scripts (8)	System call
Domain 1	225.4	557.1	18.2	861.9	374.5	216.0	46.5	296.4
Domain 2	152.4	357.3	6.7	609.5	292.7	156.7	26.5	199.0
Comparison (%)	67.6	64.1	36.8	70.7	78.2	72.5	57.0	67.1

Table 2

Test								
VM	Arithmetic	Dhrystone	Execl	File copy (1024B)	Pipe throughput	Process creation	Shell scripts (8)	System call
Domain 1	379.8	917.6	596.9	1555.2	661.7	392.0	1589.3	496.8
Domain 2	3.9	9.6	8.3	15.8	6.5	3.6	11.2	4.8
Comparison (%)	1.03	1.05	1.40	1.02	0.98	0.92	0.70	0.97

same time on the same hypervisor system. Xen keeps all the ChWall-types of running VMs. When a VM is to be started, resumed or migrated-in, Xen checks whether the ChWall-types of this VM appear together with any running ChWall-type in any conflict set. If it does, then this VM is not allowed to be started, or resumed or migrated-in. Otherwise, there may be illegal information flow between the running VMs.

2.3.2 Attack description: In Xen, there is a global structure variable chwall_bin_pol, which has an array field conflict_aggregate_set, as shown in Fig. 4. It stores all the ChWall-types that are conflicted with any of the ChWall-types of running VMs. For instance, if there are four ChWall-types A, B, C, D. A and B are in a conflict set. When a VM with ChWall-types A is running, then the content of conflict_aggregate_set is 0,1,0,0, which means the conflicted ChWall-type is B. Hence when a VM with ChWall-type B is to be started, it will be not allowed. Attackers can modify the value of conflict_aggregate_set to make two VMs that should not run together run on the system at the same time. For instance, by changing its content all to be zero, although a VM with ChWall-type A is running, the VM with ChWall-type B can also be started.

2.3.3 Impact: In our experiments, we define ChWall-types and a conflict set as the following:

$$\text{ChWall all-types} = \{\text{SystemManagement, Sensitive, Distrusted}\}$$

$$\text{ChWall all-conflictset} = \{\text{Sensitive, Distrusted}\}$$

We assign dom0 with SystemMangement, domain 1 with Distrusted and domain 2 with Sensitive ChWall-type. After the system boots with Chinese Wall enabled, we run domain 1 first. Then we try to start domain 2, it is denied since it has conflicted types with the running domain 1. Then in domain 1, we insert a module that has DMA capability to alter conflict_aggregate_set to be zero. After that we start domain 2 again, this time it is allowed to be started. This kind of attack may make domain 1 steal sensitive information from domain 2. In cloud environment, domains that represent different companies may run on one server platform. Chinese Wall policy can be used to disallow information flow between enterprises that have competitive relationship. However, if an attacker alter such security policy related data, Chinese Wall policy will be broken down and illegal information flow between domains may happen.

3 Discussion

In this section, we first discuss the difficulty in attacks that target on hypervisor non-control data. Then we discuss current approaches that protect hypervisors to see whether they can be used to defend hypervisors from non-control data attacks.

In our experiment, we implemented attacks on certain version of the Xen (4.1.2) hypervisor and assumed that the global data and the offset of important fields in the data structure are known. In cloud environment, different versions of the Xen hypervisors may be used. The attacker needs to know the hypervisor version first and therefore understand the data layout of the underlying hypervisor. However, even with a certain version of the Xen hypervisor, the field offsets may vary owing to the configuration of the hypervisor. For instance, the offset of the is_privileged field in the Xen 4.2 version is different if the administrator configures it differently. Even if the attacker knows the address of the domain structure, he cannot decide the address of is_privileged field. Hence how to identify the offsets of data fields is still an open problem. We could learn from the work Patagonix [10], which depends on binary format specifications of executables instead of symbol information to detect covertly executing binaries without making assumptions about OS kernels. We plan to implement an identity procedure that may use data mining technology to identify the offsets before launching a non-control data attack in our future work.

Our non-control data attacks are restricted to open source hypervisors. For closed source hypervisors, such as VMware ESX server, it is difficult to implement a non-control data attacks since its data layout is not known and thus we cannot further identify what kinds of data are critical to system security. Also, in previous work that did researches on non-control data attack [4, 5], they focused on open source applications or operation systems, without targeting on closed source applications or operating system. Hence before a successful attack, the attackers need to know what kind of the underlying hypervisor is. Some hypervisor detection techniques [24–26] could be used to identify the hypervisor type. If the underlying hypervisor is the expected hypervisor, such as Xen, the attacker could launch the non-control data attack. Otherwise, he may give up or launch other types of attacks on the hypervisor.

For the defenses, there are some approaches based on code measurement to protect hypervisor code. HyperGuard [13] and HyperCheck [14] put the hypervisor measurement agent (MA) in the CPU system management mode (SMM), present in × 86 system, to isolate MA from the hypervisor it protects. Although MA can measure the code integrity of the hypervisor, its protection components are restrict since the SMM does not provide all the contextual information required by MA. HyperSentry [15] solves such problem by putting MA in the hypervisor but uses the SMM to protect it. Most of these approaches are code-based integrity measurement, which can detect attacks that alter hypervisor code or injecting malicious code. In our attack model, we use ROP or DMA technique to alter hypervisor non-control data directly without modifying any hypervisor code, so it cannot be detected by these techniques.

As for data integrity, HyperSafe [16] is designed to defend hypervisor against control-flow hijacking attack. It uses hardware features to implement a non-bypassable memory lockdown technique so that only a special routine in the hypervisor can write to the memory. It also implements more fine-grained control flow integrity [27] using a restricted pointer indexing technique. However, it focuses on control data of hypervisors so it cannot be used to protect hypervisor non-control data.

Besides the approaches that harden current commercial hypervisor, there are other approaches towards developing a new specialised prototype hypervisor. For instance, Trustvisor [28] is a thin special-purpose hypervisor that provides code integrity as well as data secrecy for an application. Bitvisor [29] is also a thin hypervisor but designed to focus on I/O device security. NOVA [30] takes a microkernel-like approach to the virtualisation level by moving most functionality to user level. All the above approaches are designed to reduce the code base of

doi: 10.1049/iet-ifs.2012.0252

hypervisors in order to minimise the attack surface, but their functionalities are affected and are not suitable to be used for commerce. Also, such approaches cannot defend hypervisor against attacks fundamentally. Hence they are not suitable for protecting hypervisor non-control data.

4 Related work

In the following we divided the related work into two categories: non-control data attacks and attacks on hypervisors.

4.1 Non-control data attacks

On the application layer, Chen *et al.* [4] demonstrate that non-control-data attacks against real-world applications are realistic. They exploit memory corruption vulnerabilities of applications, including FTP, SSH, Telnet and HTTP servers, to modify non-control data instead of control data in order to break into computer systems. It showed four types of data, including configuration data, user input data, user identity data and decision-making data are critical to application software security. On the operating system layer, Baliga *et al.* [5] present some attacks against kernel non-control data. It shows that by manipulating the zone watermarks to generate artificial memory pressure, attackers can cause resource wastage and performance degradation. It also shows by contaminating the entropy pool used by pseudo-random number generator, attackers can degrade the quality of pseudo random numbers generated. To the best of our knowledge, there is no research work that focuses on non-control data attacks at the virtualisation level. Our approach is a first step towards such work to motivate further innovation in this area.

4.2 Attacks on hypervisors

Rutkowska *et al.* [31] take advantage of DMA to overwrite parts of the main memory to insert a rootkit into the Xen hypervisor. Such rootkit makes attackers have the ability to load or unload modules in the Xen address space, similar to kernel modules mechanism. DMA attacks, which have been studied and exploited in the past, are not particularly new. Although IOMMU, a hardware technology for device virtualisation, can be used to prevent DMA attacks, the authors show several interrupt-based attacks that can circumvent IOMMU in [32]. In our approach, in order to show the importance of hypervisor non-control data, we also utilise DMA attacks. Different to Joanna's work, the target is new: they focus on hypervisor code and we focus on hypervisor non-control data. Another difference is that we utilise the recent advanced attack technique, ROP, to successfully escalate a domU's privilege. By constructing these attacks, we show which kinds of data are critical to system security, towards further research work to protect them.

5 Summary

In this paper, we demonstrate new target of attacks: non-control data on the virtualisation layer. By experimentally constructing such attacks, it shows the whole system performance will be affected by changing the value of such data field from one to another. Such kinds of data are vulnerable to ROP or DMA attacks. We also discuss current approaches that protect hypervisors, which mainly focus on protecting code or control data integrity of hypervisors. They are not suitable for non-control data. Our work is to motivate further research work in this area, that is, how to efficiently protect hypervisor non-control data.

6 Acknowledgments

Our work was supported by National Science and Technology Major Project No. 2010ZX01036-001-002, 2010ZX01037-001-002, the Knowledge Innovation Key Directional Programme of Chinese Academy of Sciences under Grant No. KGCX2-YW-174 and No. KGCX2-YW-125.

7 References

1 Designer, S.: '"return-to-libc" attack' (Bugtraq, 1997)
2 United States Computer Emergency Readiness Team. 'Technical Cyber Security Alerts', http://www.uscert.gov/cas/techalerts/, accessed August 2012
3 Microsoft Security Bulletin, http://www.microsoft.com/technet/security/, accessed August 2012
4 Chen, S., Xu, J., Sezer, E.C., Gauriar, P., Iyer, R.K.: 'Non-control-data attacks are realistic threats'. Proc. Int. Conf. on USENIX Security Symp., Baltimore, MD, USA, July 2005, pp. 177–192
5 Baliga, A., Kamat, P., Iftode, L.: 'Lurking in the shadows: identifying systemic threats to kernel data'. Proc. Int. Conf. IEEE Symp. on Security and Privacy, Oakland, CA, USA, May 2007, pp. 246–251
6 Fu rootkit. http://www.rootkit.com/project.php?id=12, accessed August 2012
7 Garfinkel, T., Rosenblum, M.: 'A virtual machine introspection based architecture for intrusion detection'. Proc. Int. Conf. on Annual Network and Distributed Systems Security Symp., San Diego, CA, USA, February 2003, pp. 191–206
8 Jiang, X., Wang, X., Xu, D.: 'Stealthy malware detection through vmm-based 'out-of-the-box' semantic view reconstruction'. Proc. Int. Conf. ACM Conf. on Computer and Communications Security, Alexandria, VA, USA, Oct 2007, pp. 128–138
9 Payne, D., Carbone, M., Sharif, M., Lee, W.: 'Lares: an architecture for secure active monitoring using virtualization'. Proc. Int. Conf. on IEEE Symp. on Security and Privacy, Oakland, CA, USA, May 2008, pp. 233–247
10 Litty, L., Lagar-Cavilla, H.A., Lie, D.: 'Hypervisor support for identifying covertly executing binaries'. Proc. Int. Conf. on USENIX Security Symp., San Jose, CA, USA, July 2008, pp. 243–258
11 Barham, P., Dragovic, B., Fraser, K., *et al.*: 'Xen and the Art of Virtualization'. Proc. Int. Conf. ACM Symp. on Operating Systems Principles, Bolton Landing, NY, USA, Oct 2003, pp. 164–177
12 CVE-2011–1898. http://cve.mitre.org/cgi-bin/cvena-me.cgi? name = CVE-2011–1898, accessed August 2012
13 Wojtczuk R., Rutkowska J., Tereshkin A.: 'Xen 0wningTrilogy'. http://invisiblethingslab.com/itl/Resources.html, accessed August 2012
14 Wang, J., Stavrou, A., Ghosh, A.: 'HyperCheck: a hardware-assisted integrity monitor'. Proc. Int. Conf. Int. Symp. on Recent Advances in Intrusion Detection, Ottawa, Ontario, Canada, September 2010, pp. 158–177
15 Azab, A.M., Ning, P., Wang, Z., Jiang, X., Zhang, X., Skalsky, N.C.: 'HyperSentry: enabling stealthy in-context measurement of hypervisor integrity'. Proc. Int. Conf. ACM Conf. on Computer and Communications Security, Chicago, IL, USA, October 2010, pp. 38–49
16 Wang, Z., Jiang, X.: 'HyperSafe: a lightweight approach to provide lifetime hypervisor control-flow integrity'. Proc. Int. Conf. IEEE Symp. on Security and Privacy, Oakland, CA, USA, May, 2010, pp. 380–395
17 Shacham, H.: 'The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)'. Proc. Int. Conf. ACM Conf. on Computer and Communications Security, Alexandria, VA, USA, October 2007, pp. 552–561
18 Ding, B., Wu, Y., He, Y., Tian, S., Guan, B., Wu, G.: 'Return-oriented programming attack on the Xen hypervisor'. Proc. Int. Conf. Int. Conf. on Availability, Reliability and Security, Prague, Czech, August 2012, pp. 479–484
19 http://www.tux.org/pub/tux/ benchmarks/System/unixbench, accessed August 2012
20 Coker, G.: 'Xen security modules (xsm)'. Proc. Int. Conf. on XenSummit, New York, USA, April 2007

21 Wright, C., Cowan, C., Smalley, S., Morris, J., Kroah-Hartman, G.: 'Linux security modules: general security support for the linux kernel'. Proc. Int. Conf. on USENIX Security Symp., San Francisco, CA, USA, August 2002, pp. 17–31
22 Sailer, R., Valdez, E., Jaeger, T., *et al*.: 'shype: secure hypervisor approach to trusted virtualized systems'. IBM Research Report RC23511, 2005
23 Brewer, D.F.C., Nash, M.J.: 'The chinese wall security policy'. Proc. Int. Conf. on IEEE Symp. on Security and Privacy, Oakland, CA, USA, May, 1989, pp. 206–214
24 Fraser K.: 'x86: Update xen-detect utility to scan for Xen signature in CPUID space'. December 2008, xen-unstable mailing list
25 Liston T., Skoudis E.: 'On the cutting edge: Thwarting virtual machine detection'. July 2006
26 Smith, V., Quist, D.: 'Hacking Malware: offense is the new defense'. Proc. Int. Conf. Defcon, Las Vegas, NV, USA, August 2006
27 Abadi, M., Budiu, M., Erlingsson, U., Ligatti, J.: 'Control-flow integrity: principles, implementations, and applications'. Proc. Int. Conf. ACM Conf. on Computer and Communications Security, Alexandria, Virginia, USA, November, 2005, pp. 340–353
28 McCune, J.M., Li, Y., Qu, N., *et al*.: 'Trustvisor: Efficient tcb reduction and attestation'. Proc. Int. Conf. IEEE Symp. on Security and Privacy, Berleley/Oakland, CA, USA, May 2010, pp. 143–158
29 Shinagawa, T., Eiraku, H., Tanimoto, K., *et al*.: 'BitVisor: a thin Hypervisor for enforcing I/O device security'. Proc. Int. Conf. ACM SIGPLAN/SIGOPS Int. Conf. on Virtual Execution Environments, Washington DC, March 2009, pp. 121–130
30 Steinberg, U., Kauer, B.: 'NOVA: A microhypervisor-based secure virtualization architecture'. Proc. Int. Conf. on ACM European Conf. in Computer Systems, Paris, France, April 2010, pp. 209–222
31 Rutkowska, J., Wojtczuk, R.: 'Preventing and detecting Xen hypervisor subversions'. Proc. Int. Conf. on Blackhat, Las Vegas, NV, USA, August 2008
32 Wojtczuk R., Rutkowska J.: 'Following the white rabbit: software attacks against Intel VT-d technology'. Available at http://invisiblethingslab.com/resources/2011, accessed August, 2012

doi: 10.1049/iet-ifs.2012.0252