White Paper*
Protecting e-Government Against Attacks

Gernot Heiser
NICTA† and University of New South Wales
Sydney, Australia
gernot@nicta.com.au
February 2013

1 Scope

E-Government services operate, by definition, across the Internet: citizens use their own desktops or mobile devices to access, via the Internet, government services hosted on servers physically located in some government agency, or even on a private or public cloud.

Attacks on e-government can such be broadly divided into three categories: *server-side attacks* (i.e. on the government servers), *client-side attacks* (i.e. on the citizen's computing/access device) and *network attacks* (i.e. on the Internet connection, either by interfering with existing connections/sessions or by an attacker pretending to the server to be a valid client or to the client to be a valid server). This analysis explicitly *ignores network attacks*, as these are outside our expertise.

2 Attack Surface

Almost all software is faulty. Software systems are incredibly complex, they are the most complex artefacts built by humans by orders of magnitude! Such complexity makes mistakes inevitable.

Typical defect rates in software that has gone through industry-standard quality assurance are of the order of several defects (bugs) per thousand lines of code (kLOC). Real-world software systems consist of many millions of lines of code, and hence have thousands of bugs.

Not all bugs are threats to security: many are innocuous in that they cannot be exploited by an attacker. However, in a security-critical part of the system, we have to estimate that of the order of 10 % of bugs constitute security vulnerabilities which can be exploited by an attacker, given the right circumstances.

Hence, if the critical system measures a million lines, we have to assume that it has literally *hundreds, if not thousands, of vulnerabilities*. This is the system's *attack surface*.

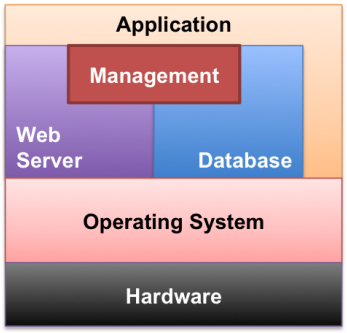

Figure 1: Traditional server software stack.

3 Server-Side Vulnerabilities

3.1 Operating system attacks

Modern computer systems consist of layers of software: increasingly specialised software components are "stacked" on top of the hardware (which forms the bottom layer).

A (fairly course-grained) view of a traditional server environment is shown in Figure 1: the bottom-most (and most general) layer is the *operating system* (typically Windows or Linux), on which there runs a *web server* and a *database management system*, the service-specific *application software*. In reality, some of the components may run on separate hardware platforms, although there is always an operating system between the hardware and the other software.

In general, the damage that can be caused by a security compromise (i.e. successful attack) is higher if it happens on lower levels of the stack: usually, if a particular layer is compromised, all layers above are trivially compromised as well. Therefore, the operating system, as the lowest layer, is the primary target of attacks.

†NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

How is the operating system attacked? In theory, the operating system is protected from higher software level by a number of technical means, which are all based in some simple hardware mechanisms. Of course, the hardware can be faulty, but that is usually a very low risk. The predominant vulnerabilities are bugs in the operating system, which can be triggered by a higher software layer when invoking an operating-system service. Modern operating systems comprise tens of millions of lines of code, which translates into thousands of vulnerabilities – a big attack surface (see Table 1).

Component	Total size	Critical Part	Vulnerabilities
Management	1 MLOC	1 MLOC	100s
Web server, database, application	10 MLOC	1 MLOC	100s
Operating System	10 MLOC	10 MLOC	1000s
Hypervisor	1 MLOC	1 MLOC	100s

Table 1: Typical code sizes and attack surface (conservative ballpark figures).

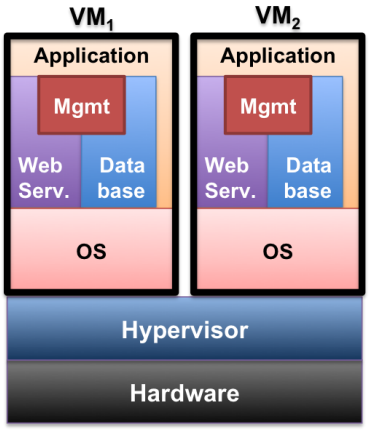

Figure 2: Servers in virtual machines (VMs).

3.2 Hypervisor attacks

Modern servers typically have an additional layer between the operating system and the hardware: the *hypervisor* (Microsoft Hyper-V, Citrix XenServer or VMware ESX). This provides multiple *virtual machines*, each appearing like actual hardware and running a traditional software stack, from operating system to application (see Figure 2). It allows *consolidating* multiple systems (complete, independent servers) onto a single hardware platform.

Hypervisor-based virtualisation has revolutionised the server world, enabling improved resource utilisation (and thus reduced capital and running cost), and are the key enabler of cloud computing and "green computing". They are here to stay.

However, the hypervisor adds to the attack surface of the system, as shown in Table 1. And, as the lowest level of software, it is the biggest prize to claim: a successful attack on the hypervisor constitutes a complete compromise of the whole system, the hypervisor can access everything.

In particular, the hypervisor enables a new class of attacks: server-to-server. The threat scenario is sketched in Figure 3. If one virtual machine is compromised, and carries out a successful attack on the hypervisor, it will compromise all the other virtual machines running on the same hypervisor. Plenty of hypervisor compromises, across all major vendors, have been reported recently, so this threat is real.

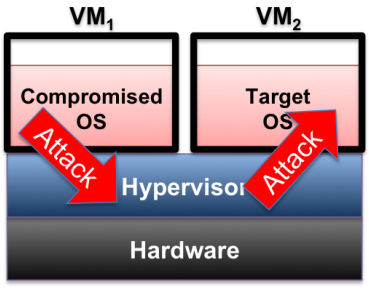

Figure 3: Attacking a virtual machine by compromising the hypervisor.

Hypervisor attacks are particularly dangerous, as the whole point of the hypervisor is to provide isolation between servers: it creates the illusion that the servers run on separate hardware (where they are strongly isolated from each other), while in reality they run on the same computer. For that reason, server-to-server attacks are often not thoroughly considered in a threat analysis.

Furthermore, part of the resource management enabled by hypervisors is the migration of services between hardware platforms. Therefore it often cannot be foreseen which services get co-located on the same physical machine. This greatly complicates any threat assessment.

Hypervisor attacks are usually launched from compromised operating systems. This seems to minimise the risk, as two vulnerabilities need to be exploited (operating system and hypervisor). However, vulnerabilities in main-stream operating systems are so plentiful that this is not much of a limitation in practice. Plenty of hypervisor compromises, across all major vendors, have been reported recently.

3.3 Side-channel attacks

As if this wasn't scary enough, hypervisors enable a particularly sneaky form of an attack, which goes through the hypervisor without actually compromising the hypervisor itself: so-called side channels. These

are communication channels which "should not exist" in the sense that they are not part of the visible mechanisms provided by a system. They are, in a sense, well-hidden loopholes.

For example, a program running within a virtual machine can observe its own progress compared to the passing of real (wall clock) time, and from that learn about other activities on the system.

Side-channels though the hypervisor have been successfully employed to steal encryption keys from a co-located virtual machine – they are real. Note that the theft of an encryption key compromises more than a single session or client: it has the potential to make *all* clients' data accessible.

Protection against side channels is particularly difficult, as they are not based on what's commonly considered "bugs", in particular, they are not violations of functional specifications: A hypervisor could be functionally perfect, in the sense that it always performs exactly the correct operation in any circumstance, yet still allow the theft of data. Discovering and closing these channels requires highly sophisticated analysis of all possible (unintended and unspecified) side effects of hardware and software mechanisms. A complete analysis in totally unfeasible for systems of the complexity of commercial hypervisors.

4 Client-Side Vulnerabilities

The citizen's access device is also prone to attacks. Typically, this is a desktop computer or a mobile device, hereafter called *terminal* (because from the server's point of view, the connection terminates there).

Terminal vulnerabilities have the same fundamental reasons as the server's: software complexity and the vulnerabilities inherently resulting from that. Furthermore, terminals are usually not professionally managed, and a wealth of applications software, much of it from completely untrusted sources, is installed. As a result, a large fraction of terminals are already compromised (typically by viruses and worms).

Client-side vulnerabilities are in a sense worse than server-side ones, as they are more plentiful, and harder to control by administrative or policy means. Fortunately, they are also more constrained in the damage they can cause, as a compromised terminal will, in general, only affect its owner's use of e-Government (involuntary participation in botnets notwithstanding). However, e-Government is doomed if citizens cannot have a reasonable degree of trust their terminals.

5 Possible Solutions

5.1 Microkernel architecture

As the core problem is the large attack surface of modern systems, any real solution must aim to reduce this. Reducing the overall size of the software stack is unrealistic; to the contrary, this will continue to grow.

Reducing the defect density may be possible, but in any case will take decades, if developments over the last 30 years provide any guidance: software defect densities have not decreased dramatically.

So, the only hope rest on *reducing the size of the critical components*, especially of the lower layers. Reducing the attack surface of the hypervisor must be the primary aim.

The key is changing the *architecture* of the system, so that the amount of critical code (that must be trusted to operate correctly) is minimised. So, instead of a "monolithic" design, the operating system or hypervisor is constructed from a minimal "microkernel", with the actual system services provided by isolated components running on top of the kernel.

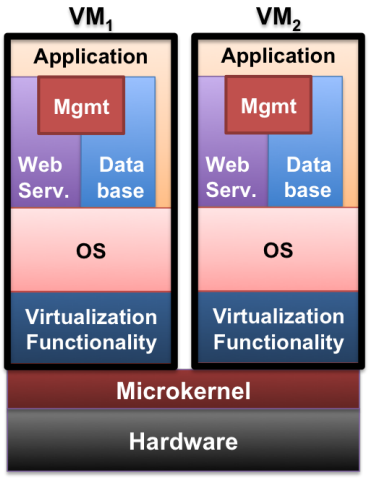

Figure 4: Splitting virtualization functionality into a microkernel and per-VM component reduces attack surface.

This means that compromises can be contained in individual components, as long as the kernel itself is not compromised. In the case of a hypervisor, most of its functionality is distributed with the individual virtual machines, as shown in Figure 4. In this case only the microkernel and some of the management components are most critical in the sense hat compromising them will compromise *all* virtual machines. This architecture is supported by NICTA's seL4 microkernel [KEH+09] as well as the Nova microkernel from TU Dresden [SK10].

5.2 Real-world microkernels

The author and his team at UNSW and NICTA have a 15-year track record of building high-performance microkernels, and are generally considered the leaders in the field, particularly with respect to high assurance as well as practical deployment. The author's startup company Open Kernel Labs (recently acquired

by General Dynamics) has deployed *billions* of copies of their OKL4 microkernel [OKL12], so this technology is clearly ready for the real world.

A modern, well-designed microkernel comprises only about 10,000 LOC. This means that, even with traditional means, the number of defects (and vulnerabilities) can be expected to be in the dozens or less.

Moreover, the small size makes it possible to employ mathematical proof techniques to show that a microkernel is *free of implementation bugs*, i.e. implements the specified functionality correctly – the ultimate dependability property. The author's team has achieved exactly that a few years ago with their seL4 microkernel [KEH+09]. In the meantime, the team has extended these proofs to showing that seL4 has the required properties for building secure systems. To date, seL4 is still the only operating system or hypervisor with this degree of assurance.

A remaining issue is that of side channels, these are not covered by the correctness proofs of seL4 (although more recent NICTA work is making significant progress). The argument here is that the small size of the microkernel makes it feasible to perform a complete analysis of side channels, eliminate most and understand the remaining ones and the threats they pose in security.

5.3 Servers

Re-architecting large legacy systems is hard. For addressing server vulnerabilities, the hypervisor is the most important piece which must be changed to a microkernel-based structure. Fortunately, there is hope, as shown by recent research results: Researchers at the University of British Columbia, Citrix and the NSA have recently shown that it is possible (albeit difficult) to break the Xen hypervisor into smaller pieces [CNZ+11], although this is still a far cry for a small microkernel.

Particularly interesting is work done by researchers at Fudan University, China, who showed that it is possible to install a small microkernel *underneath* the Xen hypervisor, so that a compromised Xen can no longer compromise the virtual machines running on top [ZCCZ11]. This is a perfect approach for eliminating the hypervisor's large attack surface by running a microkernel like seL4 underneath.

5.4 Terminals

The key to protecting terminals is, maybe surprisingly, also virtualization. Open Kernel Labs and Motorola have shown as far back as four years ago that it is possible to use a hypervisor, even on a mobile phone [Hei09b]. While this was originally done for other reasons, several companies are now using virtualization to provide secure communication on essentially off-the-shelf phones.

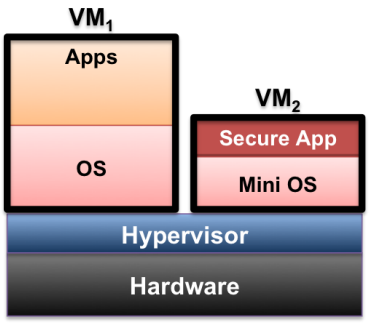

Figure 5: Smartphone virtualization.

These designs provide a secure virtual machine for the critical communication software, while a separate virtual machine runs the normal smartphone software (see Figure 5). The owner can use this virtual machine without restrictions, but its compromise is prevented to spread to the secure side, provided that the hypervisor is not compromised.

Note that such security advantages cannot obtained by some of the commercial virtualization solutions targeting smartphones, such as VMware's Horizon Mobile. This technology adds virtualization functionality into the existing smartphone operating system, rather than in a hypervisor running underneath the operating system. It is therefore just as suceptible to attacks as the existing operating system (and possible more) [Hei09a].

6 Recommendations

While we have outlined some approaches for addressing threats to e-government, these are not yet readily deployable. Much exists as *technology* (as contrasted to *products*), some is still in the research state. Productization of secure virtualization technology is not proceeding rapidly as there is no established market. Government policy can help by creating more certainty.

Recommendation 1: State a requirement for mandatory use of secure virtualization technology in e-government servers within, say, 5 years. This will provide industry with the incentive to deliver appropriate products.

At present, seL4 is the only technology which is able to provides truly trustworthy virtualization, and does so without significant performance overhead. However, seL4 is privately owned (by General Dynamics, through their acquisition of Open Kernel Labs). There is the obvious danger of government policy creating a private monopoly.

Recommendation 2: Fund a program aimed at producing an *open-source* equivalent to seL4, with the same strength of assurance resulting from

mathematical proof, and added side-channel mitigation. This is achievable with a two-year project in which NICTA would be happy to participate.

The above initiatives should be the key to addressing the server-side challenges. The technology for addressing client-side security exist, but the trend in the smartphone sector is presently towards low-security products, resulting from a perceived lack of a market for secure solutions.

Recommendation 3: Require that after, say, 5 years, e-government services will only be accessible from terminals with certified secure communication components (as can be achieved with microkernel-based virtualization technology).

Acknowledgements

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[CNZ+11] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, P. Loscocco, and A. Warfield. Breaking up is hard to do: Security and functionality in a commodity hypervisor. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, Cascais, Portugal, October 2011.

[Hei09a] G. Heiser. Hey VMware: Secure it ain't! `http://microkerneldude.wordpress.com/2011/09/09/hey-vmware-secure-it-aint/`, September 2009. Author's blog.

[Hei09b] G. Heiser. The Motorola Evoke QA4: A case study in mobile virtualization. White paper, Open Kernel Labs, July 2009. `http://www.ok-labs.com/_assets/image_library/evoke.pdf`.

[KEH+09] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: Formal verification of an OS kernel. In *ACM Symposium on Operating Systems Principles*, pages 207–220, Big Sky, MT, USA, October 2009. ACM.

[OKL12] Open Kernel Labs software surpasses milestone of 1.5 billion mobile device shipments. `http://www.ok-labs.com/releases/release/ok-labs-software-surpasses-milestone-of-1.5-billion-mobile-device-shipments`, January 2012. Media Release.

[SK10] U. Steinberg and B. Kauer. NOVA: A microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th EuroSys Conference*, pages 209–222, Paris, France, April 2010.

[ZCCZ11] F. Zhang, J. Chen, H. Chen, and B. Zang. CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, pages 203–216, Cascais, Portugal, 2011.