Hypervisors as a Foothold for Personal Computer Security: An Agenda for the Research Community

Matei Zaharia
Sachin Katti
Chris Grier
Vern Paxson
Scott Shenker
Ion Stoica
Dawn Song

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-12
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-12.html

January 13, 2012

Acknowledgement

This research is supported in part by gifts from Google, SAP, Amazon Web Services, Cloudera, Ericsson, Huawei, IBM, Intel, Mark Logic, Microsoft, NEC Labs, Network Appliance, Oracle, Splunk and VMWare, by DARPA (contract #FA8650-11-C-7136), and by the National Sciences and Engineering Research Council of Canada.

Hypervisors as a Foothold for Personal Computer Security: An Agenda for the Research Community

Matei Zaharia, Sachin Katti[†], Chris Grier, Vern Paxson, Scott Shenker, Ion Stoica, Dawn Song
University of California, Berkeley, [†]Stanford University

1 Introduction

The current consumer software stack makes end-user systems extremely difficult to secure. Consumer operating systems are large and complex, so they are easily subverted by malware, which makes its way onto users' machines either by exploiting vulnerable applications or through social engineering. Once malware has compromised the OS, it can easily disable security applications running in the OS. Afterwards, the malware is free to engage in information theft or to enlist the user's machine in attacks on third parties. As a result, there is a flourishing market in compromised bank accounts, credit cards, login credentials, and botnets [14].

Developing solutions that both significantly improve security and can feasibly be deployed is a major challenge for researchers and OS vendors. Reengineering operating systems is too costly to be practical, trusted hardware is difficult to deploy widely in the near term, and mechanisms that work within the current software stack invariably get compromised.

In this paper, we argue for an approach that leverages the growing trend of virtualization: significantly improve end-user security through a hypervisor for personal computers. Two factors motivate this approach. First, as a small programmable layer between the OS, the network, and the hardware, the hypervisor is a uniquely attractive insertion point for security. Second, hardware virtualization support has been available in desktop and laptop CPUs for several years, and lowers the overhead of virtualization below the point where most users would notice. Hypervisors thus represent one of the best opportunities to sidestep the vulnerable consumer software stack.

Previous work has explored security uses for hypervisors in isolating applications [11], protecting the OS from rootkits [22], and reducing the impact of bots [12]. However, we believe that there is a broader opportunity to directly improve user security without severely degrading the user experience. To illustrate, we discuss several hypervisor-based primitives that protect users' interactions with online services (*e.g.,* banks) from both malware and phishing. These include a secure remote UI that hides the entire interaction from the OS and user authentication and secure input primitives that can be called from existing applications, including web applications. We also sketch how hypervisors can inform users when their machine is infected by malware and aid in cleaning it up, both of which are pain points for users.

The purpose of this paper is to propose the creation of a security-enhancing hypervisor for PCs as a collaborative agenda for the research community. This agenda is not necessarily about answering fundamentally new research questions. Rather, it is a call to action about a rare chance for the community to have substantial impact. If researchers demonstrate compelling near-term benefits from a modest security layer, then OS vendors may adopt such a layer as a way to increase security without costly reengineering. The introduction of this secure foothold into the consumer software stack could then yield significant long-term benefits by providing a much better avenue for deploying security solutions.

This agenda consists of two parts: (1) exploring how hypervisors can address end-user security issues and (2) exploring how to architect a small, secure hypervisor that provides several of these facilities. We believe that there are interesting and worthwhile challenges in both parts.

The rest of this paper is organized as follows. We begin by explaining why hypervisors provide a highly attractive insertion point for security (§2) and summarizing work in this area (§3). We then discuss security facilities that a hypervisor can provide in §4, with a focus on trusted paths to online services. We conclude by discussing challenges associated with our proposal in §5.

2 Why Hypervisors?

We believe that hypervisors are uniquely positioned as a security platform because of their combination of tractability, capabilities, and incentive alignment with OS vendors, chip vendors and users.

Hypervisors are tractable: Hypervisors have always been small enough for research teams to build [7, 6]. However, hardware virtualization support allows hypervisors to become drastically simpler, obviating the need for binary rewriting and software page table management. For example, the NOVA microhypervisor [21], designed to solely use hardware virtualization, has a 9000-line trusted computing base yet outperforms commercial hypervisors. This decrease in size should also make hypervisors easier to secure. At this size, it may even be possible to formally verify hypervisors, as was recently accomplished for the L4 microkernel [16].

Hypervisors have relevant capabilities: The goal of this project is not to prevent computers from being compromised; the complexity of modern OSes and the gullibility of most users makes compromise too easy to reliably prevent. The goal, instead, is to allow compromised machines to still perform some useful tasks without sacrificing user information, and to mitigate the damage that compromised hosts can do to others. Hypervisors are well equipped for this, having direct access to the computer's hardware (particularly its keyboard, display, and processor), so that a compromised OS cannot interfere with the hypervisor. Hypervisors can also interpose on all communication between the OS and the network.

Incentives align: A hypervisor-based security platform aligns well with the incentives of operating system vendors, chip vendors and users. OS vendors would appreciate a way to improve security that does not require costly reengineering of their operating system code. If a hypervisor-based security platform built by the academic community provided an adequate proof-of-concept, one could imagine these vendors shipping such a hypervisor (not the one built by the academic community, but one built by the vendor) as part of their OS package.

Chip vendors have expressed strong interest in improving security (*e.g.,* the CEO of Intel has declared security to be "job one" [1]). It is difficult for chips themselves to make great strides in security without changes in software. However, virtualization provides a good example for what might happen with security. Initially, hardware vendors had little interest in virtualization, but once VMware and Xen demonstrated the market for virtualization by achieving significant deployment, and provided a stable target for acceleration, chip vendors rushed to add support for this feature. Similarly, if a market for a hypervisor-based security platform were demonstrated, chip vendors could add more value to their chips by finding ways to improve its security and its performance.

Finally, users are conscious of the impact of malware and appreciate unintrusive solutions that boost security. For example, gamers are even buying hardware tokens to protect their World of Warcraft accounts [5]. Building a hypervisor platform that protects online interactions might lead to wide enough deployment that other hypervisor-based security ideas, such as limiting bots' ability to send spam [12], can also begin to have impact.

3 Related Work

A substantial body of work on hypervisor-based security already exists. This work broadly falls into four classes:

- **Isolation:** At its most basic level, virtualization can be used to run applications that do not need to interact in different VMs, which is attractive in datacenter environments [11]. It has also been used to isolate intrusion detection and antivirus tools from the OS they monitor, as in Livewire [10] and VMsafe [3].
- **Desktop software stacks based on multiple VMs:** Terra [9] is a trusted hypervisor that runs both general-purpose "open-box" VMs and tamper-proof, attested "closed-box" VMs for specific applications, such as DRM-enforcing media players, cheat-proof games, or "trusted access points" for enforcing network access policies. Lampson and others proposed running separate VMs for sensitive and non-sensitive data [17].
- **OS and application protection:** By monitoring interactions with the hardware, hypervisors can protect running OSes against attacks. For example, SecVisor [20] ensures that only approved code can execute in kernel mode, HookSafe [22] protects kernels against rootkits, and Overshadow [8] protects application data against a compromised OS.
- **Limiting the impact of bots on third parties:** Not-a-Bot [12] diminishes the ability of bots to send spam and launch DoS attacks by providing human activity attestation (*e.g.,* attesting that a person was at the keyboard when an email was sent).

Many of these ideas can be incorporated into the hypervisor security platform we have proposed, but we believe that many other opportunities to leverage hypervisors arise when the focus is shifted to home users. Specifically, we urge the community to design solutions that directly help the user (rather than third parties), that proactively bypass malware (as opposed to disabling some types of attacks), and that are minimally intrusive (and hence easier to deploy). To illustrate, we present several primitives that protect users' interactions with online services. We believe that if researchers design a hypervisor security platform that users themselves want and that OS vendors find sufficiently beneficial, then this platform could also facilitate the deployment of other useful security mechanisms that are not compelling enough by themselves to achieve wide deployment, such as mechanisms to limit the ability of bots to send spam.

There are also serious systems challenges in architecting a hypervisor that can perform *multiple* of these proposed security functions while retaining a small trusted computing base. We discuss these challenges in §5.

4 Security Facilities for a PC Hypervisor

In this section, we present several ways in which hypervisors can directly address security concerns for end-users. We spend most of our attention on securing interactions with online services to prevent information theft. We also sketch ideas on leveraging hypervisors to monitor systems for malicious activity and to remove malware.

Lastly, we list some previously proposed security functions that are well-suited for inclusion in a hypervisor for personal computers.

4.1 Protecting Interactions with Online Services

Establishing a trusted path between an online service and its users is a fundamental problem in Internet security, critical for applications that perform financial transactions or process sensitive data. The problem goes well beyond man-in-the-middle attacks and key loggers. An equally important issue is phishing attacks that simply masquerade as a given website. Distressingly, most users overlook or misunderstand browser security indicators [19]. The combination of masquerading and malware is especially worrisome, because malware can control everything the user sees. Some malware already alters online bank statements shown by users' browsers to hide its transactions [2]. Lastly, a third set of problems arise from the use of passwords for authentication: users often pick weak passwords or reuse them across sites [13].

Hypervisors can play two roles in establishing a trusted path to online services. First, they can read input from and deliver output to the user in a way that bypasses the OS, provided that phishing concerns are dealt with. Second, we propose using hypervisors to store keys that act as authentication factors for accessing online services, similar to hardware tokens or two-factor authentication using mobile phones. Unlike external authentication factors, this approach is immune to man-in-the-middle attacks where a phisher tricks a user into typing a code onto a phishing site, because the hypervisor authenticates itself directly to the online service. Furthermore, it does not require the user to carry extra hardware.

These capabilities can be combined in several ways to provide various degrees of security. We discuss three such primitives in the rest of this section:

- *Authentication only:* This is sufficient for applications where theft of login credentials is the main concern, such as online games and social networks.
- *Secure input prompts:* This mechanism allows both websites and native applications to initiate hypervisor-controlled dialogs for sending sensitive input to a remote server (*e.g.,* credit card numbers).
- *Secure remote UIs:* For the most sensitive applications, such as online banking, the hypervisor can completely hide the interaction from the OS. We discuss a minimal and general mechanism for achieving this.

Hypervisor-Assisted Authentication: The user authentication schemes commonly used today are vulnerable to various combinations of phishing, man-in-the-middle attacks, and malware. Passwords can be intercepted by all three of these types of attacks, and are also often weak. Two-factor authentication schemes using external devices (such as hardware tokens) solve some of these problems, but require the user to carry an extra device. In addition, under these schemes, both phishing sites and malware can still hijack user sessions.

We propose a hypervisor-assisted authentication scheme that gets around many of these issues. In this scheme, each user is identified to each online service by a key specific for that service. The hypervisor keeps these keys in storage inaccessible to the OS. Users receive their keys either on USB drives or over the Internet if the user registers the service through a hypervisor-controlled UI (using the remote UI primitive described later); in both cases, the key does not pass through the OS. Also, the hypervisor knows which service each key is associated with, identifying services through the existing SSL PKI.

When a user wishes to log into a remote application (either through a website or through a local client program such as a game client), authentication proceeds as follows. First, the client side of the application initiates a request to the hypervisor, passing it the name of the service and a nonce. This is logically done through a hypercall, although such a "hypercall" can be implemented without modifying OSes and browsers by sending a specially formed network packet.[1] The hypervisor then asks the user whether she wishes to authenticate to the service in question using a hypervisor-controlled popup dialog. If so, it contacts the service (authenticating it through SSL), presents the nonce, and answers a cryptographic challenge to prove that it has the user's key.

This scheme provides slightly stronger security properties than external authentication factors without requiring the user to carry another device. First, the user's authentication key cannot be accessed by the OS and hence by malware. Users also cannot be tricked to reveal their keys, *because they do not know them.* Second, man-in-the-middle attackers (*e.g.,* phishing sites) cannot masquerade as the user's machine because the hypervisor opens an SSL connection directly to the online service when it needs to answer the cryptographic challenge. Thus, there are only two ways for attackers to log in as the user: either obtain physical access to the user's machine, or attempt to hijack a user session through malware (*e.g.,* wait for the user to log into a bank website and then start taking actions on her behalf). The physical attack vector can be mitigated by allowing the user to set up a "local password" that needs to be typed into the hypervisor-controlled dialog to unlock a key.[2]

[1]We propose DNS requests to a nonexistent subdomain of the website of the service that the application wants to use, such as `<nonce>.auth.chase.com`. These requests can even be made by JavaScript in a web application. Since the hypervisor can see all outgoing network traffic, it can intercept such packets.

[2]Note that if an attacker obtains this local password, she *still* needs physical access to the user's machine, because there is no way for malware to spoof keyboard input into the hypervisor.

In summary, hypervisor-assisted authentication is inexpensive to deploy, usable from an unmodified OS and browser, more convenient than two-factor authentication through physical devices, and also more powerful.

Secure Input Prompts: The authentication primitive in the previous section can prevent attackers from logging in as the user on an online service, but is not enough to prevent them from viewing information the user types into the service. For example, a key logger can still capture the user's credit card number. To protect the most sensitive information entry, the scheme can be extended slightly to allow the online service to prompt the user through a hypervisor-controlled UI. Specifically, after the hypervisor authenticates the user to the service, the service can send back a question to ask in another hypervisor-controlled dialog box. User's interactions with this dialog will be hidden from the OS.

The main security challenge with this approach is UI spoofing. While malware cannot view the user's interactions with the hypervisor, it can pop up UIs that look similar to the hypervisor's dialogs at a moment when the user expects a secure input prompt. We believe that this attack can be mitigated to some extent using a "reverse password" that authenticates the hypervisor's UI to the user, such as a background texture or color scheme for hypervisor UIs that is chosen randomly on each PC and thus cannot be easily guessed by malware. This scheme is similar to SiteKey for web applications, except that there is no way for attackers to view the hypervisor UI and capture the reverse password.[3] Nonetheless, protecting gullible users from UI spoofing is still a difficult challenge that requires significant human factors research.

Secure Remote UIs: The final primitive we discuss is a fully hypervisor-controlled remote UI that hides *all* user input from the OS, as well as all output presented by the service to the user. For example, banking sites may encourage users to use this UI so that malware cannot see their bank statements and account numbers, or an enterprise may ask employees to use this mechanism for remote sessions to their work machines.

The main challenge here is designing a mechanism that is convenient for the user, has a minimal trusted codebase, and is general enough to support a wide range of remote services. One strawman approach would be to have each service distribute client software in the form of a VM, as in Terra [9]. However, this approach has three disadvantages. First, VMs consume significant resources and take a nontrivial time to start up, which may damage the user experience. Second, the VMs would need to be updated periodically with new versions of web browsers or other programs, which would be time-consuming and expensive. Third, a VM is a large amount of code to trust, so the hypervisor would most likely need to protect the user's system against rogue client VMs.

[3]We assume that the hypervisor protects graphics memory it uses.

For similar reasons, we believe that a browser is also not the best client. Browsers are smaller than VMs, but still fairly large. In addition, they are evolving provide more capabilities (*e.g.,* HTML5), so services would want users' browsers to be kept up to date. Lastly, browsers may not be a rich enough client for all applications.

Instead, we believe that the best client software for this primitive is a remote desktop client such as VNC [4]. In effect, we are proposing a virtual thin client. Remote desktop clients are not only small, but also unlikely to need frequent updates, and general enough to support many types of applications. Also, existing applications are easy to expose through remote UI servers.

4.2 Monitoring for Malware

A second user concern that we believe hypervisors can help with is letting users know when their machines are infected with malware. Today, users may only get a vague sense that their machine is infected if they are not running antivirus software. Furthermore, malware can disable such software before vendors distribute a signature for it. One previously proposed approach that helps is running antivirus tools in a separate VM [10, 3]. However, we believe that it may also be beneficial to employ behavioral monitoring in the hypervisor. For example, having the hypervisor tell the user "your computer sent 1000 emails today" is enough to raise their suspicion. This type of monitoring works even for new malware malware for which signatures have not been disseminated. This idea is not particularly deep or novel, but it is an example of how a hypervisor security platform may be useful as attackers become more sophisticated.

4.3 Disinfection

A more speculative idea that we also believe could be useful in the future is to use the hypervisor as a "last bastion" for disabling malware once a user's machine is infected. The hypervisor appears to be a very strong vantage point from which to disable malware, because it can view and alter all contents of memory and the entire instruction stream. As such, it may be possible to build recipes for disabling each strain of malware that can be disseminated to hypervisors. There may exist effective countermeasures for attackers, such as obfuscation techniques that generate very different instruction streams on each infected machine, but this appears to be harder than obfuscating against current forms of virus signatures.

4.4 Other Facilities

Existing work on protecting the kernel from rootkits [22], ensuring that only approved code executes in kernel mode [20], and preventing bots from harming third parties [12] would also be helpful to include in a hypervisor security platform. If the community rallies to build a platform that aligns with the incentives of users and OS vendors, these ideas could have significant impact.

5 Conclusion and Challenges

We argued that the research community should focus its efforts on a hypervisor-based security platform because such a system could (i) be built by the community, (ii) support a wide range of security features, and (iii) achieve widespread deployment and impact. As such, we think it represents perhaps the best chance the community has to bring significant near-term security benefits to mass-market users. Such a layer would also make security solutions far more deployable in the long term.

This agenda requires substantial work in two main areas: developing small, secure hypervisors, and exploring how they can address the security concerns of end-users. Both of these areas pose challenges and opportunities well worth the community's attention. We conclude by sketching three of the major challenges.

Architecting a Versatile Yet Secure Hypervisor: A successful hypervisor security platform will need to implement not only one or two security mechanisms, but a judiciously chosen subset out of an already large array of proposed mechanisms. It is thus critical to design such a platform in a way that ensures that none of the security mechanisms becomes vulnerable to attack itself. We believe that this will need to be accomplished through a combination of architecture (defining a set of low-level primitives that the hypervisor exposes for use by security mechanisms), isolation, and possibly formal verification.

Fortunately, there is a wealth of research into secure OSes that can be applied to this problem. One promising approach is to limit the permissions of each system: for example, a system that protects the OS against rootkits has no need to access the network. It may also be possible to have most of the security mechanisms be written in a restricted language that provides strong guarantees.

Apart from security, the design of the hypervisor will also need to take into account efficiency (*e.g.,* for features that inspect network traffic) and updatability (ideally, the hypervisor should update itself without rebooting).

Attacker Responses to Hypervisor Security: As with any new security mechanism, it is important to consider how attackers might evade hypervisor-based security solutions. For example, one natural question is how malware can evade the hypervisor-assisted detection of spam and DoS behavior described in §4.2.

Authenticating the Hypervisor to the User: Although we believe that "reverse password" schemes, as described in Section 4.1, are a useful tool in creating a trusted path between the hypervisor and the user, this path will likely remain one of the most attractive points to attack in a hypervisor security platform. Protecting users that are poorly informed, gullible, and not paying attention against UI spoofing is a difficult problem, and ultimately needs to be addressed through user studies. Fortunately, we can leverage much of the existing research on phishing prevention and secure UIs [15, 18].

References

[1] Intel stuffs speedy security into silicon. `http://www.theregister.co.uk/2010/06/17/intel_security_silicon/`.

[2] New malware re-writes online bank statements to cover fraud. `http://tinyurl.com/ylrtnyq`.

[3] VMware VMsafe security technology. `http://www.vmware.com/technology/security/vmsafe.html`.

[4] VNC. `http://en.wikipedia.org/wiki/VNC`.

[5] World of warcraft to sell token device for added security. `http://tinyurl.com/5dy555`.

[6] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP 2003*, pages 164–177.

[7] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum. Disco: running commodity operating systems on scalable multiprocessors. *ACM Trans. Comput. Syst.*, 15(4):412–447, 1997.

[8] X. Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. Ports. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. In *Proc. ASPLOS 2008*.

[9] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. *SIGOPS Oper. Syst. Rev.*, 37(5):193–206, 2003.

[10] T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *Proc. NDSS*, volume 1, pages 253–285, 2003.

[11] T. Garfinkel and A. Warfield. What virtualization can do for security. *;login: USENIX Magazine*, 32(6):28–34, 2007.

[12] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy. Not-a-bot: improving service availability in the face of botnet attacks. In *NSDI'09*.

[13] Y. Jianxin, A. Blackwell, and R. Anderson. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5):25–29, 2004.

[14] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying spamming botnets using Botlab. In *NSDI 2009*.

[15] C. Karlof, J. D. Tygar, and D. Wagner. Conditioned-safe ceremonies and a user study of an application to web authentication. In *SOUPS '09*, Pittsburgh, PA, USA.

[16] 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 *SOSP '09*, 2009.

[17] B. Lampson. Accountability and freedom. `http://research.microsoft.com/en-us/um/people/blampson/slides/AccountabilityAndFreedom.pdf`, 2005.

[18] J. A. Rome. Compartmented mode workstations. `http://www.ornl.gov/~jar/doecmw.pdf`.

[19] S. E. Schechter, R. Dhamija, A. Ozment, and I. Fischer. The emperor's new security indicators. In *IEEE Security and Privacy*.

[20] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity oses. In *SOSP '07*.

[21] U. Steinberg and B. Kauer. NOVA: a microhypervisor-based secure virtualization architecture. In *EuroSys '10*.

[22] Z. Wang, X. Jiang, W. Cui, and P. Ning. Countering kernel rootkits with lightweight hook protection. In *CCS '09*.