A Secure Cloud with Minimal Provider Trust *

Amin Mosayyebzadeh[1], Gerardo Ravago[1], Apoorve Mohan[6], Ali Raza[1], Sahil Tikale[1],
Nabil Schear[2], Trammell Hudson[3], Jason Hennessey[1],
Naved Ansari[1], Kyle Hogan[5], Charles Munson[2],
Larry Rudolph[3], Gene Cooperman[6], Peter Desnoyers[6], Orran Krieger[1]

[1]*Boston University, Boston, MA* [2]*MIT Lincoln Laboratory, Lexington, MA* [3]*Two Sigma, New York, NY*
[5]MIT, Cambridge, MA [6]Northeastern University, Boston, MA

Abstract

Bolted is a new architecture for a bare metal cloud with the goal of providing security-sensitive customers of a cloud the same level of security and control that they can obtain in their own private data centers. It allows tenants to elastically allocate secure resources within a cloud while being protected from other previous, current, and future tenants of the cloud. The provisioning of a new server to a tenant isolates a bare metal server, only allowing it to communicate with other tenant's servers once its critical firmware and software have been attested to the tenant. Tenants, rather than the provider, control the tradeoffs between security, price, and performance. A prototype demonstrates scalable end-to-end security with small overhead compared to a less secure alternative.

1 Introduction

Despite all the advantages of today's public clouds, many security sensitive organizations are reluctant to use them because of their security challenges and the trust that the tenant needs to place in the cloud provider. Can we make a cloud that is appropriate for even the most security sensitive tenants? Can we make a cloud where the tenant does not need to fully trust the provider? Can we do this without hurting the performance of tenants that do not wish to pay for extra security ?

The key security challenge of Infrastructure-as-a-Service (IaaS) clouds stems from collocating multiple tenants on a single physical node with virtualization. Malicious tenants can exploit vulnerabilities in the huge trusted computing base (TCB) to launch attacks on tenants running on the same node [29] or even worse, if the attacker compromises the hypervisor, to launch attacks on the cloud provider. Moreover, virtualization enables side-channel and covert channel attacks such as the recent Meltdown and Spectre exploits [31, 34, 35, 44, 45]. Recent processor secure enclave technology Intel SGX has suffered from its own security challenges from the collocation of multiple tenants [10, 16, 33, 40, 52]. Such security concerns keep huge sections of the economy, such as medical companies and hospitals, financial institutions, federal agencies etc., from being able to take advantage of the benefits of today's clouds. [6, 8, 11, 42][1]

Bare metal clouds [27, 28, 41, 43, 48] remove the threat of side-channel attacks and covert channels implicit in virtualization. However, all the existing bare metal clouds still require the tenant to fully trust the provider which may not always be a safe assumption. Consider how one protects against server firmware attacks. If a prior tenant of a node is able to inject malicious firmware, this modified firmware can be used to attack future tenants of that server. Existing clouds protect against firmware attacks on the tenant's behalf [15, 27] but there is no way for the tenant to verify, for example, that the provider was not compromised or even has installed all firmware security patches.

Bolted differs from today's bare metal clouds by reducing the implicit trust in the provider. Bolted allows a tenant to elastically carve out a secure private enclave of commodity physical servers in which she may run applications. The enclave is protected from previous users of the same servers (using hardware-based attestation), concurrent tenants of the cloud (using network isolation and encryption), and future users of the same servers (using storage encryption and memory scrub-

*DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering. Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

[1] IARPA recently released RFI [7] describing the requirements of one security-sensitive community to *replicating as closely as possible the properties of an air-gapped private enclave*. We believe that meeting this requirement would alleviate the concerns of a broad community of security-sensitive customers, making the geographical distribution, elasticity, and on-demand pricing of cloud available to a wide community of users.

bing). With Bolted an organization with security expertise is able to deploy their own attestation infrastructure, and can directly validate the measurements against their expectations of firmware and software deployed. Each Bolted component addresses a different potential vulnerability. It is the combination of Bolted components that minimizes the role of the provider to mostly that of securing the physical access to the hardware.

We have implemented a prototype of Bolted and demonstrate that the prototype can provision an enclave of sixteen servers with a full application environment with legacy servers and firmware in around 8 minutes, enabling highly elastic environments. We show that the overhead to do attestation with Bolted is low, adding only 25% to the cost of a highly optimized provisioning system. Replacing traditional UEFI firmware with a customized Linux-based firmware we developed for this service, we further improve the security of the user and reduce provisioning time to being around 10% faster than the unattested version; a node can be fully provisioned with a full application environment in just over 3 minutes.

2 Threat Model

Our goal in Bolted is to enable tenants to strongly isolate themselves from other tenants while placing as little trust in the provider as possible. Specifically, we trust the provider to maintain the physical security of the hardware, so physical attacks like bus snooping or de-capping chips are out of scope. We also trust the provider for availability of the network and node allocation services and any network performance guarantees. We assume that all cloud provider nodes are equipped with Trusted Platform Modules (TPMs) [5].

We categorize the threats that the tenant faces into the following phases:

Prior to occupancy: Malicious (or buggy) firmware can threaten the integrity of the secure enclave of which the node becomes a part. We must ensure that a previous tenant (e.g., by exploiting firmware bugs) or cloud provider insider (e.g., by unauthorized firmware modification) did not infect the node's firmware prior to the tenant receiving it. Further, we must ensure that the node being booted is isolated from potential attackers until it is fully provisioned and all defenses are in place.

During occupancy: Although many side-channel attacks are avoided by disallowing concurrent tenants on the same server, we must ensure that the node's network traffic is isolated so that the provider or other concurrent tenants of the cloud cannot launch attacks against it or eavesdrop on its communication with other nodes in the enclave. Moreover, if network attached storage is used (as in our implementation) all communication between storage and the node must be secured.

After occupancy: We must ensure that the confidentiality of a tenant is not compromised by any of its state (e.g, storage or memory) being visible to subsequent software running on the node.

3 Design Philosophy

A central design principle of Bolted is to enable as much functionality as possible to be implemented by the tenant rather than by the provider for three reasons: (i) to minimize the trust that a tenant needs to place in the provider, (i) to enable tenants with specialized security expertise and requirements to implement functionality themselves, and (iii) to enable tenants to make their own cost/performance/security tradeoffs. This principle has a number of implications for our design.

First, Bolted differs from existing bare metal offerings in that most of the component services that make up Bolted can be operated by a tenant rather than by the provider. A security sensitive tenant can customize or replace these services. All the logic, that orchestrates how different services are used to securely deploy a tenant's software, is implemented using scripts that can be replaced or modified by the user.

Second, while we expect a provider to secure and isolate the network and storage of tenants, we only rely on the provider for availability and not for the confidentiality or integrity of the tenant's computation. In the most secure deployments, we assume that Bolted tenants will further encrypt all communication between the tenants' nodes and between those nodes and storage. Bolted provides a (user-operated) service to securely distribute keys for this purpose.

Third, we rely on attestation (measuring all firmware and software and ensuring that it matches known good values) that can be implemented by the tenant rather than just validation (ensuring that software/firmware is signed by a trusted party). This is critical for firmware which may contain bugs [12, 20, 24, 25, 46, 49] that can disrupt tenant security. Attestation provides a time-of-use proof that the provider has kept the firmware up to date. More generally, we attest through the process of incorporating a node into an enclave, and we can also continuously attest when the node is operating, to ensure that bugs in any layer of software (irrespective of who signed them) have not allowed malicious code to be executed.

Fourth, we have a strong focus on keeping our software as small as possible and making it all available via open source. In some cases, we have written our own highly specialized functionality rather than relying on larger function rich general purpose code in order to achieve this goal. For functionality deployed by the provider, this is critical to enable it to be inspected by tenants to ensure that any requirements are met. For example, previous attacks have shown that firmware security features are difficult to implement bug-free – including firmware measurements being insufficient [13], hardware protections against malicious devices not being in place [38], and dynamic root of trust (DRTM) implementation flaws [51]. Further, our firmware is deterministically built, so that the

tenant can not only inspect it for correct implementation but then attest that this is the firmware that is actually executing on the machine assigned to the tenant. For tenant deployed functionality, small open source implementations are valuable to enable user-specific customization.

4 Architecture

The architecture of Bolted and the sequence that a node goes through as part of being admitted to a tenant enclave is shown in Figure 1. The state of the node changes as a result of its interaction with the three main services that comprise the Bolted system. Like any bare metal offering, Bolted requires, an *isolation service* which allocates nodes and configures networks to isolate those nodes from nodes of other tenants and a *provisioning service* to provision the user's operating system and applications onto the allocated nodes. The Bolted architecture adds a third: an *Attestation Service*.

The attestation service consists of a server and a client component that runs in the firmware of the node. The server is responsible for: 1) maintaining a whitelist of trusted firmware/software measurements, 2) comparing *quotes* (hash measurements signed by the TPM) of firmware/software against the whitelist, 3) maintaining a *registry* of TPM to node mappings to verify quotes by the TPM and ensure that the quotes are coming from an expected node in the cloud, and 4) distributing keys to nodes so that they may encrypt communication between them as well as securely accessing storage.

The client component is responsible for participating the node in the attestation protocol. On boot, the hardware measures the first portion of firmware that in turn measures the remainder of the firmware and the next code (e.g., a bootloader) before it is executed. That code in turn loads, measures and then executes subsequent software, etc. These measurements are all stored in the TPM. The client obtains quotes from the TPM, i.e., cryptographically signed measurements of the firmware and software that are loaded and executed on the node. The client securely provides the quotes to the attestation server, which then matches them to its whitelist. The compromised software will not match the whitelist and the infected node will be rejected. Upon successful attestation, the verifier securely provides the node a cryptographic key that can bootstrap encrypted storage and network isolation. At this point, the node is acquired by the tenant. It is critical that client firmware and software be open, simple, verifiable, and, ideally, deterministically built to enable reliable and secure attestation.

5 Implementation

We have developed a prototype of the Bolted architecture. Here, we briefly describe the implementation components of the three main services, the node firmware we use and how these components work together.

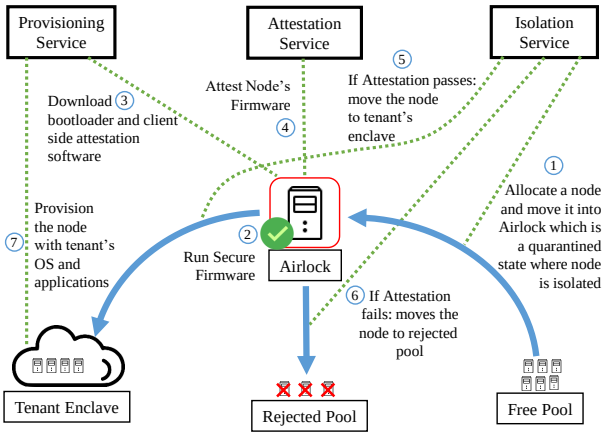

Figure 1: Bolted's architecture: Blue arrows show state changes and green dotted lines show actions taken by each service.

Hardware Isolation Layer: The Hardware Isolation Layer (HIL) [21] is our implementation of the Bolted isolation service. The fundamental operations HIL provides are (i) allocation of physical nodes, (ii) allocation of networks, and (iii) connecting these nodes and networks. A tenant can invoke HIL to allocate nodes to an enclave, create a management network between the nodes, and then connect this network to any provisioning tool (e.g., [9, 14, 17, 36]). She can then create additional networks for isolated communication between nodes and/or attach those nodes to public networks made available by the provider. HIL is a very simple service (approximately 3000 LOC). It creates networks (currently VLANs [26]) and attaches nodes to them by interacting with the switches of the provider.

Malleable Metal as a Service: The Malleable Metal-as-a-Service (M2) [36] is our implementation of the Bolted provisioning service. The fundamental operations M2 provides are: (i) create (disk) image, (ii) clone and snapshot an image, (iii) delete an image, and (iv) boot a node from an image. Similar to virtualized cloud services, M2 services images from remote-mounted boot drives. Images are exposed to the nodes via an iSCSI (TGT [19]) service managed by M2 and stored in a Ceph [50] distributed file system. As published previously, M2 is between 3-4 times faster than traditional provisioning systems that install an image into a server's local disk [36].

Keylime: Keylime [47] is our implementation of the Bolted attestation service. It is divided into four components: Registrar, Cloud Verifier, Client and Tenant. The registrar maintains the node to TPM mapping. The verifier maintains the whitelist of trusted code and checks nodes' integrity. The Keylime client is downloaded and measured by the node firmware and then passes quotes from the node's TPM to the verifier. Keylime Tenant starts the attestation process and asks Verifier to verify the node which runs Keylime client.

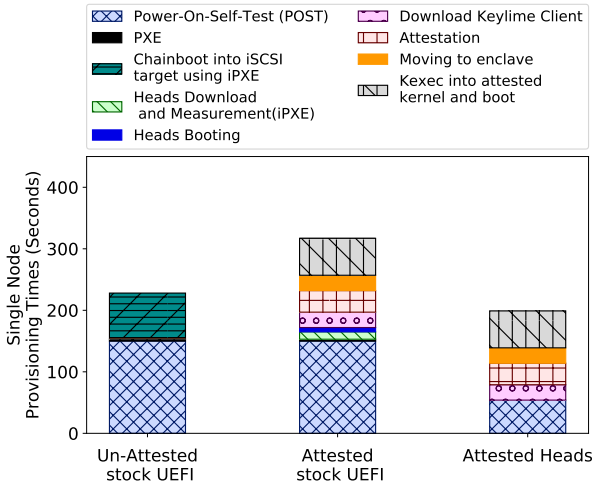

Figure 2: Performance with and without Bolted for systems with stock UEFI firmware.

Heads: Heads [23] is our firmware implementation and bootloader replacement. It is a minimal deterministically built version of Linux that (i) zeroes all node memory, (ii) downloads the Keylime client, and (if attestation has succeeded) (iii) downloads and kexecs to a tenant's kernel. In the future, we expect to directly mount the iSCSI disk from M2 to obtain the kernel, but currently, we fetch the kernel from a web service stood up for this purpose.

Putting it together: The booting of a node is controlled by a Python application that follows the sequence of steps shown in Figure 1. Secure OS images contain a Keylime client that obtains a key to encrypt network traffic between nodes in the enclave as well as traffic to the boot disk mounted using iSCSI. For servers that support it, we burn Heads directly into the server's flash, and for the other servers, we download Heads from a PXE service stood up for this purpose and then continue the same sequence as if Heads was burned into the flash. We have modified iPXE client code to measure the downloaded Heads image into a TPM register so that all software involved in booting a node can be attested.

6 Evaluation

We show performance results from our initial prototype implementation of Bolted. Timing breakdowns are shown using a Dell R630 server with 256 GB RAM and 2 2.6GHz 10 (20 HT) core Intel Xeon processors model E5-2660 v3. We have physical access to this server in our lab and show experiments with both stock UEFI firmware and our own Heads firmware. Scalability experiments are shown on 17 Dell M620 blade serves with 64 GB memory and 2 2.60GHz 8 (16 HT) core Xeon E5-2650 v2 processors connected to a 10Gbit switch.

:

Figure 2 shows the timing breakdown of different stages of provisioning with Bolted. The three scenarios are: 1) an unattested boot that just uses M2 to directly boot a users image, 2) fully attested boot with stock UEFI firmware, and 3) fully attested boot with Heads burnt into the flash.

Without Bolted's security features, a server node provisioning via M2 takes three steps: Power-on-self-test (POST), PXE, and then chainbooting from an iSCSI target using iPXE. The total time is under 4 minutes with 2.5 minutes spent in the POST step alone.

For full attestation using stock UEFI firmware, after POST, Bolted goes through the following phases: (i) PXE downloading iPXE, (ii) iPXE downloading and measuring Heads, (iii) booting Heads, (iv) download the Keylime client (currently using http) and measuring it, (v) running the Keylime client, registering the node and attesting it, and then downloading (currently using http) and measuring the tenants kernel, (vi) moving the node into the tenants enclave and, and finally (vii) Heads kexec to the tenants kernel and it is booted[2]. With all these steps the total time to provision a server is just over 5 minutes or around 25% more than the unattested boot.

For full attestation using Heads firmware, after POST we immediately jump to step 4 above. Heads posts in just under a minute (almost half of which on this server is a timeout waiting for Intel Management Engine to initialize). With Heads burned into the firmware total provisioning time is 35% faster than the fully attested case with stock firmware and even 10% faster than the unattested case with stock firmware.

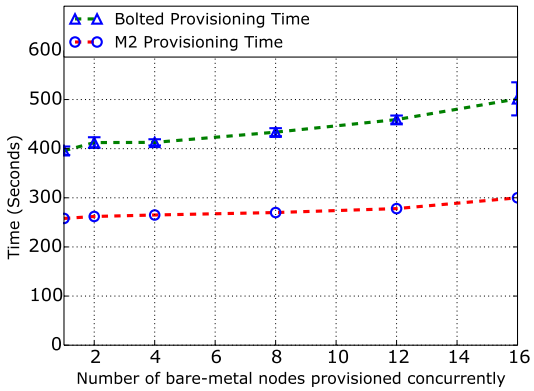

Figure 3: Initial Scaling Results.

Figure 3 shows (with UEFI firmware) the scalability of Bolted with and without attestation as we increase the number of concurrently booting nodes. Each experiment was run five times, and the line shows the degradation in performance for the average of those runs as we increase the number of

[2]This implementation is a very early prototype, and we expect to be able to speed up steps 4 and 5 by incorporating iSCSI drivers into our Heads implementation.

nodes from one to sixteen. With sixteen servers booting concurrently, performance degrades by around 50 seconds for the unattested case and around 100 seconds for the attested case. Degradation in the unattested experiments is due to a large number of concurrent block requests to the small scale Ceph deployment (with only 27 disks). Experiments [36] with a larger scale Ceph deployment demonstrated better M2 scalability. Performance of a fully attested boot degrades by only 13% as we move to 16 servers.

7 Discussion

In this paper, we describe Bolted, an architecture for a bare metal cloud that is appropriate for even the most security sensitive tenants. For these tenants, Bolted enables protection from attacks: (i) prior to occupancy using attestation to ensure that any nodes with compromised firmware are rejected and by isolating nodes (in airlocks) until they can be added to tenant enclaves, (ii) during occupancy by allocating entire servers to avoid virtualization attacks and by providing a secure model to distribute tenant keys for encrypting storage and network traffic, (iii) after occupancy by using firmware that scrubs memory prior to booting other software and using M2 for network mounted storage. The only trust these tenants need to place in the provider is: (a) the availability of the resources and (b) that the physical hardware has not been compromised.

The only Bolted service that needs to be deployed by a provider is the isolation service (e.g., HIL), which needs to be trusted by the provider to control the physical switches. All other services can be deployed by a tenant or on their behalf by a third party and the orchestration to enable an attested boot is managed by scripts controlled by the tenant. This means that a security sensitive tenant can operate these services in their own environment. Customers with specialized needs may choose to develop their own variants of these services.

The cost/complexity/performance/security trade-offs are fully under the tenants control. A tenant that doesn't want the cost and complexity to deploy their own instance of Keylime and M2, or that wants to take advantage of a large-scale implementation by the provider, can choose to trust provider-deployed versions of these services. Also, if a tenant chooses to trust the network isolation of the provider (e.g. HIL) he/she may feel no need to encrypt network and/or storage traffic. Finally, tenants that are willing to trust firmware validation (e.g. firmware is bug-free and signed by the vendor) are free to do that and will not incur any of the performance overhead of attestation.

To enable a wide community to inspect them and minimize their TCB, all components of Bolted are open source, including Keylime [4], Heads [22], M2 [2], and HIL [1]. We designed HIL, for example, to be a simple micro-service rather than a general purpose tool like IRONIC [17] or Emulab [9]. HIL is being incorporated into a variety of different use cases by adding tools and services on and around it rather than turning it into a general purpose tool. Another key example of a small open source component is Heads. Heads is much simpler than UEFI. Since it is based on Linux, it has a code base that is under constant examination by a huge community of developers. Heads is reproducibly built, so a tenant can examine the software to ensure that it meets their security requirements and then ensure that the firmware deployed on machines is the version that they require. For example, the firmware must measure all of itself before launching the next level of software. As another example, we need to make sure that firmware zeros all the memory of a server before enabling subsequent software to run[3].

We are not able to flash Heads on all servers, and the servers we have flashed it on require physical access [23]. However, we are working with the OpenCompute community to both enable Heads to be flashed remotely and to ensure that OpenCompute vendors provide Heads as a supported option; it is enormously difficult without vendor support to ensure that servers with minor changes will successfully boot. If we cannot flash our own firmware, Bolted uses stock firmware to download Heads. In this situation, Heads provides us with a standardized execution environment for the Keylime client and download the tenant kernel. While we have no guarantee that the stock firmware is up-to-date and fully measured, in this situation Bolted provides attestation against the white list of the most up-to-date firmware to ensure known vulnerabilities have been addressed.

Bolted protects against compromise of firmware executable by the system CPU; however modern systems may have other processors with persistent firmware inaccessible to the main CPU; compromise of this firmware is not addressed by this approach. These include: Base Management Controllers (BMCs) [37], the Intel Management Engine [18, 32, 39], PCIe devices with persistent flash-based firmware, like some GPUs and NICs, and storage devices [30]. Additional work (e.g. techniques like IOMMU use, disabling the Management Engine [3] and the use of systems without unnecessary firmware) may be needed to meet these threats.

While it is a work in progress, our early scalability results are encouraging, even in an initial prototype on a testbed with several known performance and scalability issues. They suggest that a complete implementation of Bolted with Heads burned into the firmware and a larger scale storage backend will enable us to elastically provision dozens, perhaps even hundreds of fully attested servers in under five minutes. If we can achieve this, it will make Bolted appropriate for highly-elastic security-sensitive situations, e.g., a national emergency requiring many computers. This will hugely reduce the need for institutions that keep around large numbers of largely idle machines to deal with low probability events.

[3] We need to zero memory in the firmware since a malicious provider may steal a server from a tenant at any point and we need to make sure that all state is removed from the node before the next tenant can see it.

References

[1] hil: Hardware Isolation Layer, formerly Hardware as a Service. https://github.com/CCI-MOC/hil.

[2] Malleable Metal as a Service (M2). https://github.com/CCI-MOC/M2.

[3] me_cleaner: Tool for partial deblobbing of intel me/txe firmware images. https://github.com/corna/me_cleaner.

[4] python-keylime: Bootstrapping and Maintaining Trust in the Cloud. https://github.com/mit-ll/python-keylime.

[5] Trusted Platform Module (TPM) Summary. https://trustedcomputinggroup.org/trusted-platform-module-tpm-summary/, Apr. 2008.

[6] 4 major reasons some organizations are still reluctant to move to the cloud|Logicalis. http://www.hypeorripe.com/2013/11/13/4-major-reasons-some-organizations-are-still-reluctant-to-move-to-the-cloud, 2013.

[7] Creating a Classified Processing Enclave in the Public Cloud|IARPA. https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/creating-a-classified-processing-enclave-in-the-public-cloud, 2017.

[8] Report to the President on Federal IT Modernization - Introduction to the Report. https://itmodernization.cio.gov/, 2017.

[9] ANDERSON, D. S., HIBLER, M., STOLLER, L., STACK, T., AND LEPREAU, J. Automatic online validation of network configuration in the emulab network testbed. In *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on* (2006), IEEE, pp. 134–142.

[10] BRASSER, F., MÜLLER, U., DMITRIENKO, A., KOSTIAINEN, K., CAPKUN, S., AND SADEGHI, A. Software grand exposure: SGX cache attacks are practical. *CoRR abs/1702.07521* (2017).

[11] BUCCI, S. Getting Cyber Serious: Mastering the Challenges of Federal Cloud Computing. *The Heritage Foundation*.

[12] BULYGIN, Y., LOUCAIDES, J., FURTAK, A., BAZHANIUK, O., AND MATROSOV, A. Summary of attacks against BIOS and secure boot. *Defcon-22* (2014).

[13] BUTTERWORTH, J., KALLENBERG, C., KOVAH, X., AND HERZOG, A. BIOS Chronomancy: Fixing the core root of trust for measurement. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, ACM, pp. 25–36.

[14] CANONICAL. Metal as a service. urlhttps://maas.ubuntu.com/.

[15] CORPORATION, I. A Trusted Cloud Solution from HyTrust, VMware, and Intel. https://www.intel.com/content/www/us/en/cloud-computing/path-to-secure-compliant-trusted-cloud-brief.html, 2016.

[16] COSTAN, V., LEBEDEV, I., AND DEVADAS, S. Sanctum: Minimal hardware extensions for strong software isolation. In *USENIX Security* (2016), vol. 16, pp. 857–874.

[17] DER VEEN ET AL., D. V. Openstack ironic wiki. urlhttps://wiki.openstack.org/wiki/Ironic.

[18] ERMOLOV, M., AND GORYACHY, M. How to hack a turned - off computer, or running unsigned code in intel management engine. https://www.blackhat.com/docs/eu-17/materials/eu-17-Goryachy-How-To-Hack-A-Turned-Off-Computer-Or-Running-Unsigned-Code-In-Intel-Management-Engine.pdf, Dec 2017.

[19] FUJITA, T., AND CHRISTIE, M. tgt: Framework for storage target drivers. In *Proceedings of the Linux Symposium* (2006), vol. 1, Citeseer, pp. 303–312.

[20] HEASMAN, J. Rootkit threats. *Network Security 2006*, 1 (2006), 18–19.

[21] HENNESSEY, J., TIKALE, S., TURK, A., KAYNAR, E. U., HILL, C., DESNOYERS, P., AND KRIEGER, O. HIL: Designing an exokernel for the data center. In *Proceedings of the 7th ACM Symposium on Cloud Computing (SoCC'16)* (Santa Clara, CA, Oct. 2016).

[22] HUDSON, T. heads: A minimal Linux that runs as a coreboot or LinuxBoot ROM payload to provide a secure, flexible boot environment for laptops and servers. https://github.com/osresearch/heads.

[23] HUDSON, T. Heads Webpage. https://trmm.net/Heads.

[24] HUDSON, T., KOVAH, X., AND KALLENBERG, C. ThunderStrike 2: Sith Strike. *Black Hat USA Briefings* (2015).

[25] HUDSON, T., AND RUDOLPH, L. Thunderstrike: EFI firmware bootkits for Apple Macbooks. In *Proceedings of the 8th ACM International Systems and Storage Conference* (2015), ACM, p. 15.

[26] IEEE. 802.1q-2014 - bridges and bridged networks. http://www.ieee802.org/1/pages/802.1Q-2014.html.

[27] INC., A. W. S. Amazon EC2 Bare Metal Instances with Direct Access to Hardware. https://aws.amazon.com/blogs/aws/new-amazon-ec2-bare-metal-instances-with-direct-access-to-hardware/, 2017.

[28] INTERNAP. Bare-metal AgileSERVER. http://www.internap.com/bare-metal/, 2015.

[29] KING, S. T., AND CHEN, P. M. Subvirt: Implementing malware with virtual machines. In *Security and Privacy, 2006 IEEE Symposium on* (2006), IEEE, pp. 14–pp.

[30] KIRK, J. Destroying your hard drive is the only way to stop this super-advanced malware. https://www.pcworld.com/article/2884952/equation-cyberspies-use-unrivaled-nsastyle-techniques-to-hit-iran-russia.html, Feb 2015.

[31] KOCHER, P., GENKIN, D., GRUSS, D., HAAS, W., HAMBURG, M., LIPP, M., MANGARD, S., PRESCHER, T., SCHWARZ, M., AND YAROM, Y. Spectre attacks: Exploiting speculative execution. *ArXiv e-prints* (Jan. 2018).

[32] KROIZER, A. Tpm and intel ptt overview. http://tce.webee.eedev.technion.ac.il/wp-content/uploads/sites/8/2016/01/AK_TPM-overview-technion.pdf, Sep 2015.

[33] LEE, S., SHIH, M., GERA, P., KIM, T., KIM, H., AND PEINADO, M. Inferring fine-grained control flow inside SGX enclaves with branch shadowing. *CoRR abs/1611.06952* (2016).

[34] LIPP, M., SCHWARZ, M., GRUSS, D., PRESCHER, T., HAAS, W., MANGARD, S., KOCHER, P., GENKIN, D., YAROM, Y., AND HAMBURG, M. Meltdown. *ArXiv e-prints* (Jan. 2018).

[35] LIU, F., YAROM, Y., GE, Q., HEISER, G., AND LEE, R. B. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy* (May 2015), pp. 605–622.

[36] MOHAN, A., TURK, A., GUDIMETLA, R., TIKALE, S., HENNESSEY, J., KAYNAR, U., G.COOPERMAN, DESNOYERS, P., AND KRIEGER, O. M2: Malleable Metal as a Service. *ArXiv e-prints* (2018).

[37] MOORE, H. A penetration tester's guide to ipmi and bmcs. https://blog.rapid7.com/2013/07/02/a-penetration-testers-guide-to-ipmi/, Aug 2017.

[38] MORGAN, B., ALATA, E., NICOMETTE, V., AND KANICHE, M. Bypassing IOMMU protection against I/O attacks. In *2016 Seventh Latin-American Symposium on Dependable Computing (LADC)* (Oct 2016), pp. 145–150.

[39] NEWMAN, L. H. Intel chip flaws leave millions of devices exposed. https://www.wired.com/story/intel-management-engine-vulnerabilities-pcs-servers-iot/, Nov 2017.

[40] O'KEEFFE, D., MUTHUKUMARAN, D., AUBLIN, P.-L., KELBERT, F., PRIEBE, C., LIND, J., ZHU, H., AND PIETZUCH, P. spectre-attack-sgx. https://github.com/lsds/spectre-attack-sgx.

[41] PACKET. The promise of the cloud delivered on bare metal. https://www.packet.net, 2017.

[42] PAQUETTE, S., JAEGER, P. T., AND WILSON, S. C. Identifying the security risks associated with governmental use of cloud computing. *Government Information Quarterly 27*, 3 (July 2010), 245–253.

[43] RACKSPACE. Rackspace Cloud Big Data OnMetal. http://go.rackspace.com/baremetalbigdata/, 2015.

[44] RAZAVI, K., GRAS, B., BOSMAN, E., PRENEEL, B., GIUFFRIDA, C., AND BOS, H. Flip Feng Shui: Hammering a needle in the software stack.

[45] RISTENPART, T., TROMER, E., SHACHAM, H., AND SAVAGE, S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security* (2009), ACM, pp. 199–212.

[46] RUTKOWSKA, J. Intel x86 considered harmful, 2015. https://blog.invisiblethings.org/papers/2015/x86_harmful.pdf.

[47] SCHEAR, N., CABLE, II, P. T., MOYER, T. M., RICHARD, B., AND RUDD, R. Bootstrapping and maintaining trust in the cloud. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications* (New York, NY, USA, 2016), ACSAC '16, ACM, pp. 65–77.

[48] SOFTLAYER. Big data solutions. http://www.softlayer.com/big-data, 2015.

[49] WAGNER, H., ZACH, D.-I. M., AND LINTENHOFER, D.-I. F. M. A.-P. BIOS-rootkit LightEater.

[50] WEIL, S. A., BRANDT, S. A., MILLER, E. L., LONG, D. D., AND MALTZAHN, C. Ceph: A scalable, high-performance distributed file system. In *Proceedings of the 7th symposium on Operating systems design and implementation* (2006), USENIX Association, pp. 307–320.

[51] WOJTCZUK, R., AND RUTKOWSKA, J. Attacking intel trusted execution technology. *Black Hat DC* (2009).

[52] XU, Y., CUI, W., AND PEINADO, M. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 36th IEEE Symposium on Security and Privacy (Oakland)* (May 2015), IEEE Institute of Electrical and Electronics Engineers.