Toward Confidential Cloud Computing

MARK RUSSINOVICH, MANUEL COSTA, CÉDRIC FOURNET, DAVID CHISNALL, ANTOINE DELIGNAT-LAVAUD, SYLVAN CLEBSCH, KAPIL VASWANI, VIKAS BHATIA

EXTENDING HARDWARE-ENFORCED CRYPTOGRAPHIC PROTECTION TO DATA WHILE IN USE

Although largely driven by economies of scale, the development of the modern cloud also enables increased security. Large data centers provide aggregate availability, reliability, and security assurances. The operational cost of ensuring that operating systems, databases, and other services have secure configurations can be amortized among all tenants, allowing the cloud provider to employ experts who are responsible for security; this is often unfeasible for smaller businesses, where the role of systems administrator is often conflated with many others.

Cloud data centers are also subject to tight physical security: the number of people with physical access is limited, and the controls on their access are stricter in a large cloud provider's data centers than on premises—often vulnerable to insider threats such as disgruntled former employees leaving with a copy of sensitive data or physical media (including on-site backups).

Cloud providers systematically encrypt data in transit (on the network) and at rest (on disks and backups) using keys associated with tenants: Even if attackers gain access to a data center, they cannot see the plaintext of tenant data unless they also manage to compromise their managed keys. This trend of increasing security in the cloud will continue; the next step is *confidential computing*, extending hardware-enforced cryptographic protection to data while in use (i.e., during computation).

TRUSTING THE CLOUD

Why would tenants take security assurances from the cloud at face value? Tenants trust their provider to different extents. Some may fully trust it to keep their data secure. Some may be concerned about other tenants, software bugs, or insider attacks (for example, from data center technicians). Some may require compliance with strict privacy regulations. Some may also doubt the provider's willingness or ability to enforce its stated security policies—guaranteeing, for example, that their data will never be used without their consent—or even fear subpoenas and other legal attacks in the jurisdictions where the cloud operates. To address these concerns, tenants increasingly expect the following:

- Minimal hardware, software, and operational TCBs (trusted computing bases) for their sensitive workloads.
- Technical enforcement, rather than just business policies.
- Transparency about the guarantees, residual risks, and mitigations that they get.

Confidential computing meets these expectations by allowing tenants to exercise full control over the TCB

used to run their cloud workloads: *Confidential computing allows tenants to precisely define all the hardware and software that has access to their workloads (data and code), and it provides the technical mechanisms to verifiably enforce this guarantee. In short, tenants retain full control over their secrets.*

In particular, confidential computing can render workloads opaque to the cloud provider because tenants can use this precise level of control to prevent access to their secrets by the hypervisor and other cloud-hosting infrastructure. This prevents attacks from the cloud fabric and its operators, and complements the more traditional security goal of protecting the cloud fabric from potentially malicious tenants.

This level of control goes beyond preventing accesses by the cloud-hosting infrastructure: it allows a tenant to specify that a particular set of secrets can be processed only by a specific code module. This capability is powerful because it can be used to design resilient systems with reduced attack surfaces. Precise control over the trust placed in confidential cloud services enables useful scenarios among multiple parties that do not fully trust one another. For example, a tenant may in turn deploy a service with strong privacy assurances for its own customers; and competing parties may jointly configure and run a multiparty cloud computation (such as data analytics or machine learning) with strong technical guarantees about the use of their pooled data.

The shift to confidential computing is part of an industrywide effort. The Confidential Computing Consortium, founded in 2019, includes Alibaba, AMD, Arm,

Google, Huawei, Intel, Microsoft, nVidia, Oracle, Red Hat, and many others. The consortium exists in recognition that transitioning to a world where confidential computing is the default for all cloud services will require significant effort at all levels of the stack, starting from the hardware and including hypervisors, operating-system kernels, and cloud services.

CONFIDENTIAL COMPUTING PLATFORMS

Let's look at trusted execution environments, dynamic implementations of such, blind hypervisors, and various platform abstractions.

Hardware foundation: trusted execution environments

At the lowest level of the stack, the hardware must be able to provide a TEE (trusted execution environment) that isolates the code and data of a given confidential workload from any other code running in a system—including code running at the highest privilege levels. The hardware must also support encryption for all of its I/O, as data flows in and out of the TEE, and be able to measure and sign the contents of the TEE to produce verifiable evidence that it is secure. This in turn requires a *hardware root of trust* to hold the platform root secrets and signing keys, and a public-key infrastructure to endorse these keys. Thankfully, these features can often be fitted into new generations of existing platforms, so they can be used in *confidential mode*, rather than requiring dedicated hardware and software.

ISOLATION. For confidentiality and integrity, software running in the TEE must be safe from snooping or

interference by other parts of the system. This may involve fencing and locking mechanisms to prevent changes to trusted code once it has been loaded and measured, and to reserve resources such as cores or memory caches for the exclusive use of a TEE to mitigate side channels. Many TEEs use encryption for either all of their state or the portion that is stored outside of trusted tightly coupled memory. Side channels, such as the Spectre and Meltdown family of vulnerabilities or differential power analysis, have the potential to pierce isolation. The full hardware/software stack for TEEs must provide defenses against any that are in scope for the threat model—for example, speculative load hardening[3] in the compiler, or secure caches[8,13] and mechanisms for secure speculation[9,14] in the hardware. Some of these defenses may be built entirely at the software level—for example, by using data trace-oblivious data structures.[5]

HARDWARE ROOT OF TRUST. Protection must be rooted in hardware; otherwise, cloud operators could easily break isolation by emulating TEEs in software. Each TEE-capable device must have a unique identity, cryptographically secured with a hardware secret. This secret may, for example, be sampled and recorded in fuse banks within the device at the end of its manufacturing process, and the corresponding public key may be harvested by the manufacturer to issue the platform certificate. TPMs (trusted platform modules) provide an early example of discrete roots of trust, with limited protection from physical attacks. Google's recently released OpenTitan core and Microsoft's Pluton subsystem are examples of more advanced, integrated hardware

roots of trust. Pluton was originally created for the Xbox One gaming console and therefore has a long history of large-scale deployment into potentially hostile environments. It is also integrated into Azure Sphere IoT (Internet of things) devices and has been licensed to major CPU vendors (AMD, Intel, and Qualcomm), where it initially provides a TPM interface and can provide the hardware root of trust for confidential computing systems.

Although hardware roots of trust are now well established, they traditionally protect someone offering a service on a device (a game publisher or, in our case, a cloud provider) from misuse, whereas confidential computing further requires they protect third parties against attackers that have physical access to the device or logical access to the nonconfidential parts of the device. As an example, a mechanism that allows firmware signed by the cloud provider direct access to the hardware secret violates the promise of confidential computing.

ATTESTATION. Attestation is the mechanism that builds the confidence into confidential computing. As with cryptographic messaging systems, confidentiality without integrity is insufficient. Being able to run software in such a way that the cloud provider cannot inspect or tamper with it is no use if you can't guarantee that it really *is* the software that you expected to run.

Attestation uses keys derived from the hardware secrets maintained by the hardware root of trust to sign evidence that a TEE is in a known-good state protected by a real hardware device. This evidence is similar to a secure boot signature: a set of measurements of the TEE, for example, the hash of the initial memory contents, and the

state of various security-critical registers. Upon receiving and verifying this evidence, a remote user or software component can be certain of the integrity of the TEE and will then typically establish an encrypted channel to deploy secrets and control computation inside the TEE.

Physical isolation is the simplest way of guaranteeing confidentiality and integrity.

Hardware advancements: Dynamic TEE implementations

Physical isolation is the simplest way of guaranteeing confidentiality and integrity—for example, by using an isolated core with a simple I/O interface. This is the route taken by Apple's Secure Element (found in iOS devices and recent Macs). This is sufficient when the code running in the TEE has compute and storage requirements that are known up front and is provided by a single vendor. It is not sufficient in a cloud environment with elastic requirements and many tenants: The cloud fabric must be able to create variable-sized TEEs, to host multiple TEEs on a single system, and to dynamically allocate/deallocate resources to TEEs.

As part of the confidential computing effort, TEEs became available on most general-purpose processors over the past few years. Arm's TrustZone provided an early TEE implementation, allowing memory to be assigned at boot time to one of two worlds—secure or normal—and allowing a small trusted kernel in the secure world to provide isolated processes. In this model, all of the secure-world components must trust the secure kernel (and secure hypervisor, if present), though they do not have to trust the normal-world's hypervisor and operating system kernel(s).

Intel's SGX (Software Guard Extensions) took this a step

further and provided the ability to create isolated regions of memory in the virtual address space of user-mode processes. Any number of these regions can be created dynamically after booting, subject to resource constraints. Code and data inside the regions are protected against software attacks by access-control checks implemented by the CPU: the hypervisor and the kernel cannot see or tamper with the data inside the regions. The regions are also protected against physical attackers by memory encryption: whenever data in use leaves the CPU caches, it is encrypted before being written back to memory.

Memory encryption overhead is low if you want only to guarantee confidentiality, and not cryptographic integrity in the presence of replay attacks from an adversary with access to the memory bus. The Xbox 360 and later models have used AES (Advanced Encryption Standard) for memory encryption since their launch and provided enough bandwidth and latency for games, some of the most demanding workloads on modern CPUs. The memory controllers of mainstream CPUs are now gaining similar functionality, including support for different keys for different memory regions. Protecting large-scale memory integrity against physical attacks is more expensive; schemes to reduce this overhead are an area of active research.

SGX's isolated memory regions are ideal for small-TCB services,[10] but using them to run full VM (virtual machine) confidentiality is challenging because they lack support for multiple address spaces and privileged and unprivileged mode separation. This led AMD to develop another type of TEE focused on VM-level isolation. AMD processors

went through a series of design iterations aimed at fully removing the hypervisor from the trust boundary. Their first step, SEV (Secure Encrypted Virtualization), automatically encrypted memory in use by VMs. Next, SEV_ES (SEV with Encrypted State) added encryption of VM register state on every transition to the hypervisor. Finally, SNP (Secure Nested Paging) provided an additional guarantee that the hypervisor cannot break memory integrity by tampering with the virtual memory mappings to execute integrity or replay attacks on a confidential VM. Taken together, these features guarantee that the hypervisor cannot read or tamper with VM state (i.e., the hypervisor is out of the TCB). Intel's recently announced TDX (Trust Domain Extensions) provides a similar set of security guarantees to SNP, and it targets comprehensive VM-level confidentiality.

Finally, several important workloads require specialized processors, such as GPUs, FPGAs (field-programmable gate arrays), and other accelerators. These devices can also be augmented with TEE capabilities, with a large design space.[12] For example, some accelerators may have large memories (e.g., high-bandwidth memory) protected with physical packaging, making encryption less relevant. In many cases, accelerators can be allocated to a single tenant at a time, which removes attacks and further simplifies their designs.

To use these devices securely and efficiently, I/O buses also require changes. Most current systems assume the I/O bus is trusted, but this has been a problem for embedded systems: Until very recently the CAN (controller-area network) bus used in most cars did

not perform any end-to-end authentication, allowing compromised components such as a media player to send messages pretending to be from the engine-management system. Similarly, the PCI (peripheral component interconnect) bus specification assumes that all endpoints are trusted, enabling a malicious device to spoof the originator ID, for example, but confidential accelerators need a mechanism for establishing end-to-end secure channels between device and host TEEs and for integrating encryption between the device and the host.

Exposing confidential computing hardware requires changing the systems layer of the cloud fabric.

Virtualization advancements: Blind hypervisors

Exposing confidential computing hardware requires changing the systems layer of the cloud fabric. This includes changing the hypervisor to handle the constraint that it cannot see VM state. Mainstream hypervisor designs follow a hierarchical trust model. The hypervisor is fully trusted by the guest, is responsible for storing guest state between context switches, and has full access to guest memory.

Most paravirtualized device interfaces are designed on the assumption that any guest memory can be used for the virtual equivalent of DMA (direct memory access) buffers. These assumptions stopped being true on mainstream hardware with the introduction of AMD's SEV. Memory encryption meant that the hypervisor had to provide a region for bounce buffers that the guests could use. Linux supports this mode of operation with SEV via the software IOTLB (input/output translation lookaside buffer) driver.

When performing an explicit domain transition from inside the TEE to the surrounding environment, hardware

implementations typically preserve the register context. This is not usually true for asynchronous exits, which may leak sensitive information. In a model where the hypervisor is trusted, this information may be useful for handling the VM exit. When the hypervisor is untrusted, either it must be modified so it does not take advantage of the feature, or a shim layer must be added to sanitize the information.

For example, consider a page fault in second-level address translation. In a conventional system, the hypervisor receives a trap with the full register context and an exception register specifying the fault address. It can then either kill the VM, issue an upcall to ask the VM's kernel to handle it, or page in the missing page from backing store. In a confidential computing system, this would allow the hypervisor to single-step execution and see the register state at every step. At a minimum, the hypervisor must receive only an encrypted and integrity-protected version of the register state, but even knowing the fault address may leak too much information. In a more secure design, a shim layer inside the TEE would receive the asynchronous exit and decide whether to issue a hypercall to fill in the missing page or simply notify the kernel of the fault. This code can then enforce policy related to the frequency of such exits in order to mitigate possible attacks from the hypervisor.

It is worth noting that it is possible to create a form of TEE by trusting the hypervisor when hardware isolation is not available. This is the basis for Windows' virtualization-based security, introduced with Windows Server 2016 and Windows 10, where critical components run isolated from the Windows kernel by Hyper-V. This can support the same

abstractions as other TEEs, including isolated memory regions similar to SGX, but with a weaker threat model: in this design, the hypervisor is still trusted. This is similar to the approach recently used by Amazon's Nitro Enclaves.[1]

Platform abstractions: Confidential VMs, confidential containers, enclaves

The systems layer exposes confidential computing hardware to developers and users through a set of platform abstractions: confidential VMs, confidential containers, and enclaves.

The systems layer exposes confidential computing hardware to developers and users through a set of platform abstractions.

Confidential VMs allow tenants to have a fully backward-compatible VM experience running existing unmodified applications. In the background, systems record and check attestations to verify the security guarantees and make them auditable. Placing entire VMs in TEEs is important for fast and easy adoption, but it also causes some problems. For example, the administrator for the VM has full read/write control over the VM, which is too coarse in many cases. Another concern is that the TCB for a VM is large: a VM image is far more than just a kernel and an application; it includes a large number of system services. In the worst case, this is still likely to be more secure than running the software on premises or on existing cloud infrastructure, but there could be a better solution.

Confidential containers allow tenants to have a finer degree of control over the TCB and to run new or existing containerized applications confidentially. Over the past few years, containers have emerged as a common way of deploying software in the cloud. The exact technology

In an ideal world, software written for security would focus on a minimal TCB.

varies, but a container is typically a small (often layered or virtual) file system that contains the minimum required to run a single program.

Best practices recommend running a single microservice in each container. Orchestration infrastructure then supports deploying fleets of cooperating microservice containers. This has several advantages: the container can be configured to have a smaller TCB than a complete confidential VM, and the confidential container may run in a VM without the VM administrator being able to access it. The TEE providing the isolation for the container may still be based on VM-level isolation mechanisms (SNP, TDX, and so on), or it may be based on process-level isolation. (Systems such as Haven[2] and SGX-LKL[11] [Linux Kernel Library] adopt ideas from the library operating system and Exokernel world to run a Windows or Linux library operating system inside SGX memory regions.)

Container deployments of microservices introduce complex attestation issues: Orchestration frameworks need to provision these services with sufficient state so they can acquire keys, and they need to manage protocols for establishing the identity of an entire set of microservices.

In an ideal world, software written for security would focus on a minimal TCB. To give developers full control over the TCB, confidential computing platforms expose an enclave abstraction. Enclaves are fully flexible. They can hold as little or as much code as a developer wishes to put in them—for example, they can hold a single function that processes credit-card information or a secret signal-

processing algorithm.

As with containers, the actual hardware-level isolation mechanism for enclaves may be VM- or process-level isolation. For example, VM-level isolation can be used to create a sidecar VM that exposes a simple enclave interface to the base VM. Developers can design services that partition code with different privileges into distinct enclaves. Each enclave should be limited to access only the data necessary to perform its function. Following this principle of least-privilege requires developers to do the additional work of refactoring services, but it yields the highest security and most resilient applications.

There is still a large design space for the interfaces that enclaves will present. The OpenEnclave SDK,[6] originally from Microsoft and now part of the Confidential Computing Consortium, offers C and C++ environments, providing a relatively easy starting point for small-TCB development targeting several types of confidential hardware. Other SDKs have begun to emerge for memory-safe languages such as Rust.

CONFIDENTIAL COMPUTING APPLICATIONS

Confidential computing makes it possible to outsource sensitive workloads to the cloud, and it even introduces new computing patterns. For example, it enables secure and confidential multiparty computation in which groups of users, who may be mutually distrusting, can run joint computations and share their results without revealing their private inputs to one another or to anyone with physical or logical access to the hardware on which the computations execute. This should lead to the

development of a broad range of confidential-computing applications—in fact, these properties are already resulting in advances in multiple application domains.

Confidential AI

Machine-learning algorithms are rapidly increasing their demand for more computation and larger data sets. At the same time, their applications raise significant security and privacy concerns. The elastic and scalable nature of the cloud is already a natural choice for this type of computation, but confidential computing makes it feasible to leverage the cloud with strong security assurances.

In medical and pharmaceutical applications, for example, training data may include individuals' private health-care records, and the resulting models may be used to make clinical decisions. Running the training process inside an enclave ensures that the data cannot be viewed or modified by anyone else. It also provides integrity guarantees (and a robust audit log) that the intended training algorithm was run on the specified records with a specified software stack. As a special case of these guarantees, the resulting model may be encrypted, signed, and equipped with key release policies to ensure it will be unlocked only within another enclave that will enforce specific access control and usage restriction. The enclave may, for example, enforce differential privacy by limiting the number of times the model is queried and adding noise to their results.

As an example, a tenant can leverage confidential cloud computing to offer a medical diagnostics service to its own customers, with technical assurances that (1) it

cannot steal or misuse a model supplied and maintained by a specialist third-party for this purpose; and (2) it cannot access any personal medical information to query, store, or use the model for any other purpose.

Confidential databases and analytics

Database systems store and process sensitive and business-critical data such as personal records, financial information, and government data. Unauthorized access to such data can have serious consequences, including physical harm and loss of customer trust and competitive advantage. Current database systems provide sophisticated access-control mechanisms such as role-based access control. These mechanisms are limited in effectiveness against stronger attackers such as those with administrative or physical access to the servers. In a common example, the individual or outsourced team responsible for managing the database represents a large insider threat to a company. Because this individual or team is managing the database system, they are able to see all confidential data, even if they have no need to access it.

Over the past few years, the notion of encrypted databases has been proposed as a way of enforcing stronger, cryptographic access control. In an encrypted database, data remains encrypted both at rest and during computation, using keys that are not available even to the database or server administrator. Data appears in cleartext only within trust boundaries defined by data owners.

Encrypted databases can be realized in multiple ways.

One approach (proposed by CryptDB and other related systems) is to encrypt data on a trusted client using partially homomorphic encryption schemes. An alternative approach is to decrypt, process, and re-encrypt sensitive data within a trusted execution environment (such as Intel SGX enclaves). This approach was proposed by EnclaveDB,[7] and a related approach has been adopted by Microsoft's SQL Server in the Always Encrypted feature. The designs vary from streaming columnar data into enclaves at the time of processing to placing all sensitive data and queries within enclaves. Approaches based on TEEs have the potential to provide stronger security guarantees such as integrity and freshness of query processing, confidentiality for queries, tamperproof auditing, etc. They are also more flexible (i.e., they permit any kind of computation and can support complex access-control policies such as attribute-based access control). In addition to data processing, a TEE is a natural choice for enforcing and auditing fine-grained key-release policies.

Confidential multiparty collaboration

Secure and confidential multiparty computation enabled by enclaves[5] also facilitate new types of collaboration between data set owners, leading to a multiplicative increase in the amount of training data. Hence, instead of being limited to data from a single hospital, multiparty models can be trained on a joint data set from multiple hospitals, without revealing the constituent data sets (typically subject to complex regulations and commercial considerations). Although it may be possible to train similar models without pooling their data sets, using, for example,

federated learning or local differential privacy, these alternatives would involve algorithmic changes, additional resources, and utility losses. Scenarios for confidential multiparty collaboration exist in many other domains, including finance, energy, climate study, and government.

Confidential ledgers

General-purpose applications that run in a cloud data center need a way to convince their users that they are running correctly. New frameworks have emerged to help developers build this new class of trusted applications. These frameworks provide a simple way of building trusted applications that run inside of TEEs and produce verifiable ledgers of their execution. For example, the CCF (Confidential Consortium Framework)[4] provides a tamperproof ledger and transactional updates on a key-value store. This is used to build a number of cloud services such as Azure Confidential Ledger, which provides a tamper-evident, high-performance, confidential ledger that applications can use to store general-purpose log records for auditability and verifiability. These properties allow for a new class of applications that require coordination between mutually distrusting parties, as well as applications that require verifiable execution for legal reasons.

Trusted applications commonly use TEE attestation along with secure messaging channels. This gives users confidence that they are connecting to the correct application in a secure and confidential manner. CCF, and other comparable frameworks, additionally distribute and replicate the execution of application logic to ensure liveness and integrity of execution, even when a fraction of

Usability is critical to any security technology that aims for widespread adoption.

the nodes in the system are malicious or compromised. In particular, CCF supports Byzantine fault tolerance, where arbitrary malicious behavior is tolerated, as well as crash fault-tolerance configurations.

FOUNDATIONAL SERVICES

Usability is critical to any security technology that aims for widespread adoption. It is easy to imagine a simple configuration switch that lets tenants turn on "confidential computing" for their existing workloads, but what does it mean? Encrypting a VM, for example, adds some defense in depth but is largely security theater unless coupled with a trustworthy mechanism for attesting its contents: if the provider controls the initial VM, and the tenant has no mechanism to review it, then the provider is still fully trusted.

To get meaningful end-to-end protection, the workload inputs and outputs must be encrypted, and the associated data keys must be released only to the TEE allocated for the workload. This involves keeping track of the platform and code that are authorized for this workload, while enabling updates for both platform and code. For convenience, most tenants will likely choose to delegate these tasks to cloud services. For security, these services must themselves be deployed in TEEs, relying on one another for core functionalities such as secure identity, keying, and logging.

The next section outlines the services at the core of a confidential computing ecosystem in support of tenants that deploy confidential workloads and application developers who contribute their code.

Key management and attestation services

If a confidential service needs access to any persistent data (for example, a VM disk image) then it needs to retrieve the key from somewhere. For traditional client-side disk encryption, this can be stored in the TPM and unlocked by the tenant entering a passphrase. In a cloud scenario, the storage part is relatively easy to solve either with managed HSMs (hardware security modules) via a service such as Azure Key Vault or a persistent confidential computing service that manages key storage. Asking the tenant to enter a passphrase for every VM, container, or other service that they launch, however, is not a scalable solution.

When tenants grant access to their data, they are making a policy decision based on a review of the information included in the attestation and other metadata at their disposal. This process can be automated by a confidential cloud service, such as the Microsoft Azure Attestation: at the start of the confidential computation, the newly created TEE establishes a secure connection to the attestation service and presents its attestation materials. The service checks them against the authorization policies previously uploaded by the tenant and, if successful, issues the corresponding credentials. The TEE may then use these credentials to access tenant data. It may, for example, present a token issued by the attestation service to obtain the current decryption key from an HSM.

The cloud provider runs the attestation service atop a fleet of authorized TEEs, and then a tenant can do the manual setup step (the equivalent of entering a

passphrase) once and provide policies and credentials so that the service can automatically authorize thousands of VMs on its behalf. Within the cloud, for example, the service may automatically authorize migration between TEEs and recovery from their encrypted checkpoints.

To this end, the service maintains an up-to-date, consistent cache of platform certificates for all TEEs provisioned in its data centers. It performs frequent checks for certificate revocations and can manage, for example, the consistent deployment of firmware or microcode updates for the trusted hardware (typically requiring new collections of certificates). Thus, the service can support precise, stateful policy statements of the form, "This task must run within an SGX enclave, on an Intel SGX v2.1 platform, deployed in the German Azure data center, in a VM allocated to the tenant, supported by certificates that are valid as of today," rather than just, "This task must run within an enclave."

From a cloud perspective, it is important both to minimize the number of distinct hardware offerings and to ensure software portability across everything.

From a cloud perspective, it is important both to minimize the number of distinct hardware offerings and to ensure software portability across everything. X86 VMs will happily work on Intel or AMD hardware even though the virtualization extensions on both are different. If confidential VMs needed to be implemented differently between AMD and Intel hardware, this would add a significant barrier to adoption. By factoring out the hardware-specific details, the attestation service enables other services to be platform independent. For example, it enables the integration of legacy HSMs without the need to customize them for different forms of confidential enclaves, VMs, and containers.

Code Transparency

Remote attestation enables tenants (or services they trust) to authenticate the platform and software TCB for their computations, but it does not ensure that this TCB is trustworthy. To this end, tenants would ideally gather and review the security of all the source code for their applications, runtime SDKs and libraries, and compilation tool chains. They would then rebuild the software image for their workload and check that its hash matches the one presented for attestation. This task is daunting for several reasons:

- It involves a lot of software from different origins, even for simple applications.
- Parts of this software may be proprietary or confidential, hindering its review.
- Modern build systems are sensitive to details in their environment, yielding irreproducible code.
- Cloud computing encourages agile development and facilitates seamless code updates, often without requiring a restart of the application.
- Emergency patches may be required to mitigate newly disclosed vulnerabilities, and they can hardly wait for a security review of every impacted application.

To facilitate this task, or at least amortize its cost, a code transparency service can securely record the software dependencies, build environments, and resulting binaries used for confidential computing. A cloud provider may operate this service as a confidential ledger that systematically enforces code-update policies, signs the resulting binaries, and maintains a public, immutable log of all its operations. Accordingly, the tenant may configure the

cloud attestation and key management services to authorize any such signed binaries for a confidential application.

For example, a tenant may configure the code transparency service to install emergency patches automatically from a reputable software vendor, provided they are correctly signed and published on the release branch of their designated GitHub project. (More conservatively, they may also require signed endorsements from the cloud provider or a designated expert.) Even if the patch is broken, the service will at least provide a strong audit trail to enable its review after the fact. The software vendor might still be able to undermine a tenant's security guarantees but cannot do so without placing its reputation on the line. The code transparency service can also be used to mitigate software supply chain attacks, because it provides auditable provenance and chain-of-custody for a software bill of materials (SBoM).

CONFIDENTIAL COMPUTING IS THE FUTURE OF THE CLOUD

Confidential computing provides strong security assurances in the cloud by empowering tenants to remotely control the TCBs for their workloads: it makes explicit the (minimal) hardware, software, and services that they still need to trust; and it provides strong technical protection against any attacks from the rest—preventing potential attacks from other tenants and even from the cloud provider. This enables tenants in turn to develop and deploy their own confidential applications for their most sensitive data.

Confidential cloud computing is in its infancy, but we

believe that it will eventually become ubiquitous, the same as other privacy and integrity mechanisms such as TLS and encryption at rest. The hardware for confidential computing (CPUs, FPGAs, and accelerators) should evolve at a rapid pace, as the result of a large, concerted effort across the industry aiming to provide open, robust, standardized, platform-agnostic capabilities. As these become available, they will form the foundation for a cloud fabric that will be compartmentalized with small TCBs, where each component will have access only to the information necessary to perform its function.

This foundation will support running confidential VMs, as well as higher-level platform services. While some services may never run in a fully confidential mode, they will benefit from the new, more secure foundation. As software-engineering tools evolve toward enabling compartmentalized confidential-by-default development, these guarantees will be easy to add.

Imagine a future in which end users have complete and verifiable control over how their data is used by any cloud service. If they want their organization's documents to be indexed, a confidential indexing service could guarantee that no one outside their organization ever sees that data. A confidential videoconferencing service could guarantee end-to-end encryption without sacrificing the ability to record the session or provide transcripts, with the output sent to a confidential file-sharing service, never appearing unencrypted anywhere other than the organization's devices or confidential VMs. A confidential email system could similarly protect privacy without compromising on functionality such as searching or authoring assistance.

Ultimately, confidential computing will enable many innovative cloud services, while allowing users to retain full control over their data.

References

1. AWS Nitro Enclaves. AWS; https://aws.amazon.com/ec2/nitro/nitro-enclaves/.
2. Baumann, A., Peinado, M., Hunt, G. 2014. Shielding applications from an untrusted cloud with Haven. *Proceedings of the 11th Usenix Symposium on Operating Systems Design and Implementation*; https://www.usenix.org/conference/osdi14/technical-sessions/presentation/baumann.
3. Carruth, C. 2018. Speculative load hardening. LLVM Compiler Infrastructure; https://llvm.org/docs/SpeculativeLoadHardening.html.
4. Confidential Consortium Framework. GitHub; https://github.com/microsoft/CCF.
5. Ohrimenko, O., et al. 2016. Oblivious multi-party machine learning on trusted processors. *Proceedings of the 25th Usenix Security Symposium*, 619-636; https://dl.acm.org/doi/10.5555/3241094.3241143.
6. Open Enclave SDK. GitHub; https://github.com/openenclave/openenclave.
7. Priebe, C., Vaswani, K., Costa, M. 2018. EnclaveDB: a secure database using SGX. *Proceedings of the IEEE Symposium on Security and Privacy*; https://ieeexplore.ieee.org/document/8418608.
8. Qureshi, M. K. 2019. New attacks and defense for encrypted-address cache. *Proceedings of the 46th International Symposium on Computer Architecture*,

360-371; https://dl.acm.org/doi/10.1145/3307650.3322246.
9. Sakalis, C., et al. 2019. Efficient invisible speculative execution through selective delay and value prediction. *Proceedings of the International Symposium on Computer Architecture*; https://www.researchgate.net/publication/333755760_Efficient_Invisible_Speculative_Execution_through_Selective_Delay_and_Value_Prediction.
10. Schuster, F., et al. 2015. VC3: trustworthy data analytics in the cloud using SGX. *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, 38-54; https://dl.acm.org/doi/10.1109/SP.2015.10.
11. SGX-LKL. GitHub; https://github.com/lsds/sgx-lkl.
12. Volos, S., et al. 2018. Graviton: trusted execution environments on GPUs. *Proceedings of the 13th Usenix Symposium on Operating Systems Design and Implementation*; https://www.usenix.org/system/files/osdi18-volos.pdf.
13. Werner, M., et al. 2019. ScatterCache: thwarting cache attacks via cache set randomization. *Proceedings of the 28th Usenix Security Symposium*; https://www.usenix.org/system/files/sec19-werner.pdf.
14. Yan, M., et al. 2018. InvisiSpec: making speculative execution invisible in the cache hierarchy. *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture*; https://iacoma.cs.uiuc.edu/iacoma-papers/micro18.pdf.

Mark Russinovich *is CTO of Microsoft Azure, where he leads technical strategy and architecture for Microsoft's cloud computing platform.*

Manuel Costa *is a partner research manager at Microsoft Research Cambridge, where he leads the Confidential Computing research theme. He is interested in advancing the state of the art in security and privacy, operating systems, and programming languages.*

Cédric Fournet *is a senior principal research manager in the Confidential Computing group at Microsoft Research. He is interested in security, privacy, cryptography, distributed programming, and formally verified software.*

David Chisnall *is a principal researcher at Microsoft Research Cambridge, where he works on hardware/software co-design for security. His work spans computer architecture, operating systems, compilers, and language design. He is also an active open-source developer, contributing to LLVM since 2008 and having served two terms on the FreeBSD Core Team.*

Antoine Delignat-Lavaud *received a Ph.D. in computer science from ENS Paris, working in the PROSECCO team at Inria Paris on cryptographic protocols, programming languages, and formal verification with applications to web security. He is a principal researcher in the Confidential Computing group at Microsoft Research Cambridge and is researching protocols and systems for confidential cloud services built on hardware-security guarantees.*

Sylvan Clebsch *is a senior principal research engineer at Microsoft Research Cambridge. His work includes the Confidential Consortium Framework, Verona, and the Pony programming language.*

Kapil Vaswani *is a principal researcher in the Confidential Computing group at Microsoft Research. His research focuses on building secure, robust, and transparent systems. He has pioneered work on secure databases using trusted hardware and new forms of trusted hardware.*

Vikas Bhatia *is head of product for Azure confidential computing. Prior to Azure, he led product teams in the Windows Developer Platform group, Cloud Game Streaming, Xbox One, and Visual C++ Compiler.*