New Software Abstractions for Hardware Security Technology

Manifesto

The Future of Confidential Computing

Introduction

Our society relies on software in mobile phones, cloud computing systems, personal computers and critical distributed infrastructure. While the dependency on software has exploded, our ability to make software trustworthy, secure, and dependable has not kept pace. Security vulnerabilities, ransomware, malware, software faults, and privacy compromises are regular occurrences.

Over the last decades, important security features such as different privilege levels have been widely deployed. Programming languages have been developed that avoid common low-level security issues. Yet despite these measures, the world still loses billions of euros per year due to cyber-attacks. As the pandemic has struck, employees have been forced to work from home, and now an even bigger part of our daily lives takes place in cyberspace. Unfortunately, criminals have adapted as well and "are developing and boosting their attacks at an alarming pace". They are joined by nation states and hacktivists extending physical wars into cyberspace.

Many of these problems originate from placing trust in untrustworthy code bases. Even when applications are developed carefully, they still rely on an operating system (OS) kernel with millions of lines of code that is implemented in a memory unsafe language. A single vulnerability in the OS kernel puts the security of all application code running on top at risk. Given the sheer sizes of modern software stacks, it is unlikely that these code bases can ever be fully trusted.

To confront this challenge, hardware is evolving, and modern processors include new hardware security features. For example, *secure boot* ensures that software is guaranteed to start execution in a well-defined trustworthy environment; *trusted execution environments* (TEEs) allow the hardware to protect sensitive data and computation from malicious actors; and *hardware capability support* prevents unauthorised memory accesses by software. These new hardware features, however, pose new challenges for the software stack, from low-level firmware to hypervisors and OS kernels, middleware, and all the way to applications.

For example, instead of trying to defend the complete software stack, hardware support for TEEs focuses on protecting sensitive data during computation. TEEs create *enclaves*, which contain trustworthy code and data. The OS kernel schedules enclaves, but itself is untrusted. The processor hardware verifies and maintains the integrity of enclaves and ensures that execution can only transition to them via proper entry points. The isolation properties of TEEs allow programmers to reason about the security of their applications without loss of generality. Security vulnerabilities due to programming errors will still exist, but attacks can be mitigated by the isolation guarantees of TEEs. These strong security properties make hardware security mechanisms a key measure for modern, trustworthy software.

Various hardware security mechanisms have been proposed in academia and industry, ranging from hardware implementations such as Flicker, Sancus and TyTAN, hardware support for memory capabilities, such as CHERI, to platforms that adopt trustworthy hypervisors such as TrustVisor and Fides. All mainstream hardware manufacturers, including Intel, AMD, ARM and IBM, have either already launched hardware security mechanisms or have plans in the pipeline.

The Workshop

The workshop on New Software Abstractions for Hardware Security Technology took place at the Fondazione Monte Verità, Ascona, on October 1-4, 2023. It was organised as part of the scientific meetings program of the Congressi Stefano Franscini by the Computer Science Institute of the University of Neuchâtel, Switzerland. The scientific coordination was led by *Prof. Pascal Felber* (University of Neuchâtel, CH), *Prof. Peter Pietzuch* (Imperial College London, UK), *Prof. Christof Fetzer* (TU Dresden, DE), *Prof. Rüdiger Kapitza* (FAU Erlangen-Nürnberg, DE) and *Dr Raoul Strackx* (Fortanix Eindhoven, NL), with support from *Dr André Martin* (TU Dresden, DE), *Prof. Marcelo Pasin* (HES-SO, CH) and *Dr Valerio Schiavoni* (University of Neuchâtel, CH).

Participants of the workshop

The workshop had 46 participants from several countries (Belgium, Brazil, Germany, Italy, Netherlands, Portugal, Spain, Switzerland, UK, USA) over the course of 4 days. The program featured 20 scientific presentations, breakout sessions and plenary discussions, as well as mentoring activities and short presentations for young researchers.

The workshop's focus was on the open challenges in software systems research related to new hardware security technologies. During the event, we explored new ideas and strategies for how to create abstractions, methodologies, platforms and building blocks that will enable a new generation of trustworthy and secure software systems based on

hardware security technologies. This means devising new secure applications and system software from scratch, as well as extending and securing existing legacy applications.

Our objective was **to develop a roadmap that will inspire the systems security research community and industry to build new secure software systems**. Our thinking at the workshop was done along four axes: *new services*, *new technological advances*, *new environments*, and *new threats*. The outcome of our discussions and the resulting roadmap are described in this manifesto.

Axis 1: New Services

Confidential computing platforms are starting to be widely adopted by industry. Unfortunately, some applications are difficult to realise without relying on resources outside of the platform's trust domain. This limits the security guarantees that these applications can provide.

We advocate support for the following new services as part of confidential computing platforms in the future. First, there is a strong need for a trusted time service. Almost all applications rely on some form of public key infrastructure. This includes a requirement to assess whether a digital certificate is trustworthy or whether it has expired. Trusted time could be implemented by adding trusted time sources to a distributed network. Network time protocols such as NTP may serve as a good starting point, but care needs to be taken to defend against malicious actors at the node and network levelA form of consensus may be required and platforms may need to have some guarantees related to their own performance to achieve a robust protocol.

Second, government regulation may require data to be stored at certain geographic locations. This may be because of data protection laws or client demand for extra physical security provided by the hosting environment. Cloud providers may play a vital role in offering assurances by certifying the location of machines in their data centres. Clients may be able to infer the distance to these locations by triangulation, counting the number of hops between reserved resources. The latter may require attestation of parts of the network itself, and of the properties that it can guarantee (e.g., QoS, latency, bandwidth).

Finally, further hardware support may be added to confidential computing platforms to protect enclaves from rollback attacks. Currently, replay-protected, non-volatile memory is not pervasively available. On some platforms, a hardware TPM chip enables some protection, but access is slow and wears out. Some hardware (NVIDIA Hardware Ratchet, ICE) and software (ROTE, NIMBLE, LCM) solutions have been proposed, but none have made it to deployed confidential computing platforms.

Axis 2: New Technological Advances

Confidential computing platforms provide strong security guarantees, under the assumption that trusted enclaves operate as expected. In practice, this may not be the case, and new safeguards may be needed. Through monitoring tools and better sandboxing, one may continuously monitor an enclave and detect any traces of anomalies. This requires trustworthy evidence on the potentially malicious platform. Once a compromised enclave is detected, one needs to return it to a known good state. Here, it may be challenging to define what expected behaviour is, and false positives may be flagged. In a distributed setting, consensus may be required to identify with certainty deviations from a known good state.

The fact that enclaves may deviate from a known good state also affects attestation. Attestation commonly only handles the validation of an initial remote state of a TEE against a known good value before execution starts. We consider this as no longer sufficient. Attestation must be performed continuously, and each attestation should include proof that the enclave is still operating in a known good state. It is also important to reason about the behaviour of the enclave, verifying that it upholds a specific security policy. This may be extended to other parts of the platform, including the firmware that is currently an opaque piece of code but is used to build trust in the system. In the longer term, this may also include the verification of the manufactured hardware.

Axis 3: New Environments

Today, confidential computing is mostly concentrated in cloud settings. Vendors are converging towards confidential virtual machines. This may change in the future with confidential computing applications emerging in the IoT and edge computing spaces. Especially with the arrival of ARM CCA and RISC-V (with its security extensions) becoming more widely adopted, new real-world use cases may be possible. The heterogeneity and ubiquity of confidential computing also bring challenges. First, different platforms will need to be combined such as peripherals, accelerators, client devices and powerful servers. At the moment, it is unclear how the performance and security guarantees of these different platforms can be quantified. Multiple parties will be involved in the confidential software/hardware stack. These parties may necessitate a partial order to express their trust relationships.

Second, standards for confidential computing are needed. With more abstractions for each type of confidential computing approach (e.g., confidential VM, securing minimal code base, etc.) vendor lock-in must be reduced. This may also lead to better interoperability, portability and more widespread deployment.

Third, more tool support is needed to support heterogeneous platforms. This includes compiler support to compile enclaves for these various confidential computing platforms. Having separate SDKs for each platform makes it challenging to share code between different platforms. Also orchestrators must be aware of the different platforms they are managing and can schedule confidential workloads to different execution environments. As not all security properties carry over from one platform to the next, giving developers and operators insight of these limitations, would be valuable.

Finally, different tools will be necessary to minimise the TCB. Strengthening the security of existing applications or securing applications that perform system calls can be achieved by relying on a virtual machine-based confidential computing platform. Additional security measures may be required as a defence-in-depth approach to ensure that a breach of the VM kernel does not automatically result in a breach of the application. New applications written with the principles of confidential computing in mind will be able to achieve much

stronger security guarantees by avoiding reliance on a complete OS kernel in their TCBs. With a small TCB, formal verification of these applications may finally become feasible.

Axis 4: New Threats

Confidential computing platforms can provide much stronger security guarantees than a classic layered approach, where lower layers are strictly more powerful than higher layers. They become resistant in the presence of stronger attackers on the device. Unfortunately, there are risks to such an approach. Over the past 5 years, we have seen side-channel attacks such as transient execution attacks (Spectre and Meltdown, just to mention two notable ones) causing havoc in the industry. While these are problems of a basic building block of all computing devices, confidential computing is an easy target, as it assumes a stronger attacker model. To address these problems, industry and academia should cooperate to enable a better understanding of the microarchitecture of modern processors. It is necessary to define the information that processors can be allowed to leak during execution, carefully balancing the engineering requirements of hardware and software. Formal verification of a processor model and its data leakage is required to avoid or reduce side channels in the future.

A second threat to confidential computing is its steadily increasing TCB. Whereas initial academic prototypes and Intel SGX focused on isolating security-sensitive parts of an application, more recent approaches for confidential VMs include support for UEFI (e.g., using OVMF) and a full Linux kernel. With more intense use of primitives for secure I/O inside VMs, this may further increase the attack surface. Software components such as the Linux kernel and device drivers have never been developed with the consideration that the hypervisor running underneath may launch attacks against them. It also gives users the wrong impression that existing applications can be secured trivially by running them in a confidential VM. In order to build trustworthy applications, more must be considered. We need tools that enable developers to minimise the amount of trusted code.

Third, the industry is slowly moving towards a model in which (some) trust is placed on cloud providers. This is the case when confidential computing platforms are implemented by cloud providers, either in the form of a trusted hypervisor (e.g., AWS Nitro Enclaves) or when cloud providers design their own hardware based on ARM CCA or (eventually) RISC-V. Similarly, freshness guarantees of main memory are weakened to enclave enclaves to scale to much larger memory footprints. This also places more trust in cloud providers not to

launch (sophisticated) memory replay attacks against enclaves running on their infrastructure. All of this puts the split trust model at risk in which the infrastructure provider is untrusted while achieving confidentiality, integrity and freshness of code and data as this is delegated to the hardware manufacturer.

Full homomorphic encryption is a potential alternative to confidential computing. While it is currently several orders of magnitude slower than running similar workloads in enclaves, technological advances may reduce this overhead. It is unclear, however, when and if this will happen, and both technologies may therefore co-exist. Regulation may also lead to the adoption of one technology over the other for some applications.

Finally, there are ethical concerns related to confidential computing. While such platforms can be used to provide strong security guarantees, they may also be abused to limit personal freedom, restrict freedom of speech and prevent undesirable information from being shared. In addition, malware can take advantage of the capabilities provided by confidential computing platforms, for example, to perform cache attacks from within the enclave or to steal computing cycles from users without them noticing.

Conclusion

Confidential computing architectures can provide much stronger security guarantees than a traditional approach of privilege layers. As the technology requires confidentiality and integrity properties to hold even in the face of a powerful, privileged attacker, some services are much more difficult to provide. In addition, the imminent arrival of confidential computing in new scenarios poses yet unforeseen challenges.

The objective of our workshop was to develop a roadmap that inspires the system security research community and industry to build new secure software systems. We hope that the challenges outlined as part of the previous four axes can be used as an important starting point.

Acknowledgements

This manifesto is the result of discussions and contributions from all 46 participants at the workshop: *Alex Guardini, Alysson Bessani, André Martin, Andrey Brito, Anna Galanou, Aritra Dhar, Benny Fuhry, Charly Castes, Christian Göttel, Christof Fetzer, Clement Thorens, Dominique Devriese, Edouard Bugnion, Giovanni Mazzeo, Hans Winderix, Herbert Bos, Hugo Vincent, Jethro Beekman, Jinnan Guo, Jonas Röckl, Lluis Vilanova, Marcelo Pasin, Maxim Ritter von Onciul, Mona Vij, Neelu Kalani, Nuno Santos, Onur Mutlu, Osman Sabri Unsal Uder, Pamenas Kariuki, Pascal Felber, Patrick Eugster, Peter Pietzuch, Peterson Yuhala, Pramod Bhatotia, Quoc Do Le, Raoul Strackx, Rüdiger Kapitza, Shweta Shinde, Stuart Biles, Sven Matti Schulze, Thanikesavan Sivanthi, Thomas Preisner, Titus Abele, Valerio Schiavoni, Wojciech Ozga, Yuchen Qian*.