SPOKE 1
FUTURE HPC & BIG DATA
FLAGSHIP 4:
Selection of candidate prototypes for trustworthiness, security, privacy

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

EXECUTIVE SUMMARY

Flagship 4 aims to explore innovative technological solutions enabling multi-tenancy HPC/Cloud platforms with strong security and data privacy guarantees. As part of the activities in the flagship, documented in previous deliverable D4.FL4 *Survey of state-of-the-art approaches and gap analysis*, the participants have already identified the key areas of interest for the project, including RISC-V Trusted Execution Environments, accelerator (FPGA) oriented TEE support, secure virtualization, Federated Learning, trustworthy AI, social media data analysis, numerical analysis, trusted distributed workflows, and stochastics models.
Aiming at the selection of candidate prototypes due at Month 8 as part of Milestone 5, we identified a range of key technologies and IPs that will provide the main building blocks for the prototypes developed in Flagship 4, described in this deliverable.
The selection of key technologies and IPs, in particular, has been driven by the gaps identified in D4.FL4, so as to provide effective answers to the open issues in the state of the art. The relevant gaps are also recalled in this deliverable to provide a starting point for the presentation of the key technologies explored in Flagship 4.

Flagship 4 / Deliverable 5: Selection of candidate prototypes

Elena Baralis[1], Tania Cerquitelli[1], Alessandro Cilardo[2], Iacopo Colonnelli[3], Domenico Cotroneo[2], Alessandra De Rossi[3], Luigi De Simone[2], Roberto Esposito[3], Danilo Giordano[1], Nikhil Jha[1], Marco Mellia[1], Maurizio Munafò[1], Eliana Pastor[1], Roberto Pietrantuono[2], Mirko Polato[3], Laura Sacerdote[3], Luca Vassio[1], and Cristina Zucca[3]

[1]Politecnico di Torino, { elena.baralis, tania.cerquitelli, danilo.giordano, mellia, maurizio.munafo, eliana.pastor }@polito.it
[2]Università degli Studi di Napoli Federico II, { acilardo, cotroneo, luigi.desimone, roberto.pietrantuono }@unina.it
[3]Università degli Studi di Torino, { iacopo.colonnelli, alessandra.derossi, roberto.esposito, mirko.polato, laura.sacerdote, cristina.zucca }@unito.it

April 2023

1 Introduction

Flagship 4 aims to explore innovative technological solutions enabling multi-tenancy HPC/Cloud platforms with strong security and data privacy guarantees. As part of the previous activities in the flagship, documented in D4.FL4 *Survey of state-of-the-art approaches and gap analysis*, the participants identified the key areas of interest for the project, including RISC-V Trusted Execution Environments, accelerator (FPGA) oriented TEE support, secure virtualization, Federated Learning, trustworthy AI, social media data analysis, numerical analysis, trusted distributed workflows, and stochastics models.

Furthermore, in the context of the activities herein reported, the participants have identified a range of industrial players potentially benefiting from the flagship results. Some of the large companies in the CN have expressed their interest in the activities related to the flagship, involving resource optimization, federated learning, fault recovery, and on-premise and on-cloud resource management, as well as hardware/software-level security primitives for trustworthy computing, security algorithms and protocols for confidentiality and attestation.

Aiming at the selection of candidate prototypes also due at Month 8 as part of Milestone 5, we identified a range of key technologies and IPs that will provide the main building blocks for the prototypes developed in Flagship 4, described in this deliverable. The selection of key technologies and IPs, in particular, has been driven by the gaps identified in D4.FL4, so as to provide effective answers to the open issues in the state of the art. The relevant gaps are also recalled below as a starting point for the presentation of the key technologies in the flagship.

Gap#	**Related area**	**Gap description**
Gap01	RISC-V TEEs	limited TEE support in RISC-V microcontroller-class systems
Gap02	RISC-V TEEs	limited privilege management flexibility in RISC-V microcontroller-class systems
Gap03	Accelerator TEE support	CPU-based TEEs unfit for HPC/big data
Gap04	Accelerator TEE support	no commercial solutions accelerator-based TEEs
Gap05	Secure virtualization	limited support for RPU virtualization
Gap06	Secure virtualization	limited hypervisor misbehavior detection in hardware-assisted virtualization
Gap07	Secure virtualization	difficulty of test cases for tracing CPU virtualization bugs
Gap08	Secure virtualization	trade-offs between bug detection and isolation properties
Gap09	Federated Learning	FL lacks approaches enabling the usage of ML models that are impossible (or not easy) to be trained without gradient descent
Gap10	Federated Learning	FL needs to be augmented with additional tools for provide better privacy guarantees
Gap11	Federated Learning	Privacy constraints prevent using standard HW/SW platform as deployment target
Gap12	Trustworthy AI	lack of AI explainability benchmarking methods
Gap13	Trustworthy AI	lack of trustworthy AI solutions for ranking systems and law
Gap14	Trustworthy AI	limited support for trustworthy AI with unstructured and heterogenous data
Gap15	Trustworthy AI	user privacy vs. data utility tradeoffs
Gap16	Trustworthy AI	lack of analytical models for validating trustworthy AI proposals
Gap17	Trustworthy AI	ethical concerns in data collection
Gap18	Trustworthy AI	limited real-time analysis support for internet data

Gap19	Social media data analysis	limited means to assess accuracy, completeness, and validity of large-scale social media data sets
Gap20	Social media data analysis	data privacy and ethical concerns for privacy violation in social media data collection and analysis
Gap21	Social media data analysis	need to improved AI algorithms for analyzing social media data at scale
Gap22	Social media data analysis	need for ways of integrating cross-platform data from multiple social media
Gap23	Social media data analysis	lack of support for real-time analysis of social media data
Gap24	Numerical Analysis	NA needs efficient numerical SW for approximation using hyperparameters tuning in Big Data
Gap25	Numerical Analysis	NA lacks software for approximation and classification on large graphs
Gap26	Trusted Distributed Workflows	need for zero-trust distributed workflows
Gap27	Trusted Distributed Workflows	complexity, limited portability and maintainability of enclave-based application code
Gap28	Stochastics Models	gap and approximations between real processes and theoretical processes
Gap29	Stochastics Models	need for exact algorithms for inverse first passage time
Gap30	Stochastics Models	first passage time for a Lindley process not known

Table 1: Gaps identified in D4.FL4.

2 RISC-V Trusted Execution Environments

To address the gaps identified in deliverable D4.FL4 with respect to low-end RISC-V memory protection support, particularly Gap01 and Gap02, we aim at developing architectures with built-in protection mechanisms which 1) enable shielding from physical attacks (e.g., bus tampering, peripheral attacks, power analysis/side channels [19, 16]) and/or 2) allow new types of software defenses based on dedicated hardware-level architectural extensions. Our activity in Flagship 4 will particularly address the latter opportunity, relying on the non-proprietary RISC-V Instruction Set Architecture (ISA) specification and its open-source hardware philosophy. In this specific branch of the activity, we particularly target low- to medium-end systems used in deeply embedded industrial applications, and hence we are interested in microcontroller-class RISC-V implementations. More specifically, we aim at introducing integrated defense mechanisms serving as a baseline for establishing Trusted Execution Environments (TEEs), which are suitable for resource-constrained microcontroller-class systems. TEEs are increasingly becoming a pillar in current security architectures, ranging from server-class facilities to embedded systems, with Intel SGX and ARM TrustZone being the most prominent examples of proprietary TEE solutions. They provide an isolated execution environment based upon the paradigm of trusted computing. While a few TEE solutions exist in the technical literature for application-class RISC-V implementations, such as Keystone [22] or Sanctum [10], relying on protection [38] and isolation [21] mechanisms, we pointed out that supporting a trusted environment in deeply embedded systems based on microcontroller-class cores is particularly challenging, especially for applications having some form of latency-sensitive requirements. In the project, we will aim to identify a minimum set of hardware-level support mechanisms, with limited assumptions on the available privileged modes and memory protection measures. The reference architecture will be demonstrated by extending a lightweight RISC-V core, e.g. Ibex [25], then showing that full support for isolation properties is achieved at a reasonable overhead in terms of additional hardware resources and delay.

In particular, although existing solutions, such as the enhanced PMP unit [21], enable isolation for User mode processes, these improvements are not enough to support trusted execution in microcontroller-based systems lacking a Supervisor Mode. Modern complex systems may need to run multiple privileged software components (e.g. drivers, embedded Operating Systems, Real-Time Operating Systems [44, 26, 5, 17]), which would all run in Machine mode. While locking entries is effective for isolation, it reduces the System Software capability of creating and deleting new PMP entries, preventing the dynamic management of the trusted environment. In Flagship 4, we will address these limitations by introducing new mechanisms allowing a more versatile Machine mode PMP region management, while keeping the attack surface restrained.

2.1 Proposed RISC-V MCU-based TEE solution

Along this research line, Flagship 4 aims to identify minimal architectural extensions for effective yet flexible memory isolation which, unlike [22], are implemented at the hardware level and suitable for microcontroller-class RISC-V cores. We will first precisely formalize the notion of isolation and specialize it for RISC-V processors, in order to provide a set of properties and requirements that are needed for trusted execution environment support. By *isolation* we refer to the capability of the system of segregating different software processes to prevent them from accessing or interferng with resources that they do not own, e.g. some part of a memory space, so that a breach in a software partition will not break other partitions. This implies that violating a single partition will not give access to the entire system, resulting in a reduced attack surface, because in order to break the system an attacker has to tamper with a number of partitions $N > 1$, ideally the totality of system partitions. An additional property that is desirable in a flexible TEE, albeit not supported by PMP-based isolation mechanisms, is the capability of *dynamically* manipulating system partitions. Based on the above discussion, in RISC-V we can describe the system isolation state starting from the memory partitioning into regions, described by means of the `pmpcfgx-pmpaddrx` register pair, which encodes the permissions and the address space for each region. New memory regions allocated to privileged and non-privileged software can be created and modified at any moment. If a running process is capable of accessing control and status registers, then it is capable of changing memory regions, thereby breaking isolation. But if such a modification is legal, then the system will not reach an illegal isolation state and the attack surface will not be increased. This translates into saying that editing CSRs should not allow isolation property violation by means of system provided mechanisms (e.g. PMP priority functionalities). Based on the state of the art (bidirectional isolation between privileged processes and non-privileged ones), in Flagship 4 we will introduce a set of requirements which must be satisfied in order to dynamically manage the isolation state, without decreasing the level of security:

1. A super-privileged process (i.e. trusted System Software) must exist and be capable of completely accessing system resources.

2. The super-privileged software is the only process capable of creating, deleting, and editing any privileged and non-privileged memory region.

3. Any privileged process other than the super-privileged one cannot access system resources and cannot create nor edit any memory region.

4. Bijective isolation between privileged and non-privileged software must still hold, as well as between the super-privileged process and the privileged ones.

5. If two memory regions overlap with each other, the super-privileged ones take priority over the privileged ones.

6. The super-privileged software can grant some permissions over system resources to privileged software, which can therefore access a subset of system resources.

2.1.1 TEE architecture

For the prototype foreseen in this Flagship to address Gap01 and Gap02, as identified in D4.FL4, we refer to a general architecture of a TEE-enabled system, shown in Figure 1, enabling tight integration of the TEE into the processor. In particular, for the benchmarking, we will refer to a RISC-V-based system and the privileged modes available in a typical low-end microcontroller class device. Unlike previous proposals, such as SMART [13] and TrustLite [20], the system's trusted anchor exposes functions –the RoT services– that are safely maintained into a physical module hardwired into the processor, called the Shielded Domain (SD). SD services and keys are stored in a private addressing space and can be only accessed through special instructions, enhancing the RoT security. The remaining chain of trust can be extended via software, attaching further validated software modules in a layered fashion. The design allows software libraries and applications to be isolated during their execution thanks to the RISC-V native PMP mechanism. However, the RISC-V privilege levels normally available in a microcontroller, User and Machine modes, are not enough to build fully-functional TEEs. Therefore, we assume that the PMP is extended, similar to the context extension proposed in [27], in order to create privilege level orthogonal memory partitions called Trusted Domain (TD) and Untrusted Domain (UD). A trusted privileged runtime, called Trusted Security Monitor (TSM), is responsible for the management of UD applications, i.e. the TEEs, for their communication and their interactions with the RoT, enabling protected accesses to keys, ensuring TEE isolation, integrity, and confidentiality. Our reference architecture targets a wide spectrum of microcontrollers, ranging from ultra-low-power devices with few kilobytes of memory up to high-performance multi-core microcontrollers with hundreds of kilobytes. It is thus crucial to identify the building blocks supporting a TEE, especially for very resource-constrained microcontroller-class devices.

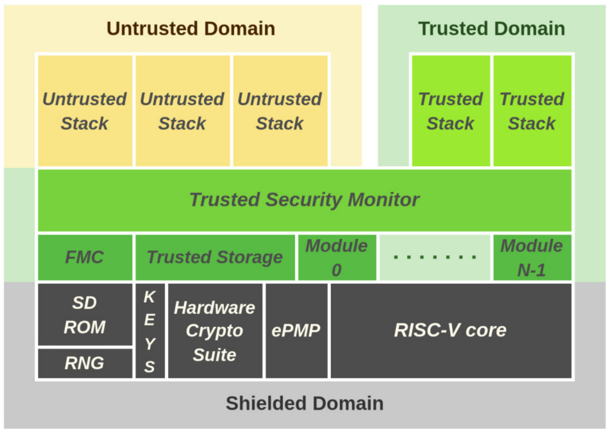

Figure 1: Reference architecture. Gray blocks denote the shielded domain, green blocks denote the trusted domain, while yellow blocks denote the untrusted domain.

Secure Boot (SB) is the process meant to ensure that the TSM only runs after the execution of trusted software modules. Starting from the reset code hardcoded into the SD ROM, the First Mutable Code (FMC), i.e. the first software module to be run, must be validated. Its Manifest, signed with the processor key called Unique Device Identifier (UDS), is read from a fixed memory region, usually flash memory, and the hashing of the software module is performed and compared to the golden values stored into the SD. In case of a match, a PMP entry is created for the FMC, and it is loaded into the address specified in its manifest and finally executed. The FMC will support new RoT services such as attestation and trusted storage. Similarly, after system peripheral initialization, the FMC may validate some new software modules. Once the TSM is brought to execution, the SB is complete.

Remote Attestation (RA) involves an external party, called the verifier, willing to verify a target device, the attester, using an attester-produced attestation report as a reference, in order to exchange information of various nature (libraries, updates etc..). Such a report is derived from the attester measurement (a hash of the attester's running software and hardware configuration) and signed with the hardware/software attestation key. Furthermore, the high degree of interconnection among IoT devices and their behaviour as a swarm suggested a new form of attestation, known as Collective Remote Attestation (CRA) [1], used to prove the genuine state of an entire group, or swarm, of devices. One of the most popular scheme is SEDA [4], which is based on a spanning tree structure used to propagate attestation requests/responses through the swarm, triggered by the peer device in the tree root being queried by an external verifier. The Flagship will also explore the integration of collective attestation mechanisms into the proposed prototype architecture.

SB and CRA primitives inherently need cryptographic hash functions, while authentication can in principle be performed based either on symmetric cryptography or public key cryptography. Given the stringent resource constraints and the relatively closed nature of a typical IoT application, making key setup/ distribution much less problematic than in a generic Internet application, we assume here that TEE primitives only use symmetric cryptography. Nevertheless, the current state of the art offers a wealth of hash and symmetric encryption algorithms, with a variety of implications on implementation efficiency.

3 Accelerator oriented TEE support

As highlighted in Deliverable D4.FL4, the joint problem of performance and privacy requirements in HPC environments poses a perspective that is fundamentally ignored in the current technological landscape, when application-specific acceleration is provided, e.g. based on GPU or FPGA accelerators. The key problems are that CPU-based Trusted Execution Environments do not directly fit the requirements of large-scale big data and HPC applications (Gap03), while special-purpose acceleration based on commercial GPU and FPGA devices do not currently provide any consistent notion of TEE for user applications (Gap04). As one of its key contributions, the PNRR CN1 Spoke 1 Flagship 4 activity will introduce a novel trusted computing paradigm for accelerated HPC/big data, demonstrated on a customized platform enclosed within the physical perimeter of a hardware-reconfigurable accelerator. Such a solution will jointly address Gap03 and Gap04. In fact, hardware reconfigurability will enable the coexistence of accelerated HPC algorithms and hardware-boosted privacy-preserving computing solutions, featuring different cryptographic mechanisms to be used for different levels of complexity of the HPC user tasks. The solution will introduce a range of new technologies, algorithms, and protocols which will

- create a notion of trusted computing in special-purpose compute platforms, particularly FPGA-based acceleration devices (used both as the final acceleration technology and as a prototyping platform);
- support hardware-boosted cryptographic solutions for privacy-preserving computing, targeting specific tasks where homomorphic encryption and garbled circuits are a good fit;
- fully preserve the potential performance of dedicated FPGA acceleration in complex HPC and big data tasks, adopting an SGX-like approach extended to the FPGA domain;
- ensure the smallest possible trusted compute base, fundamentally limited to the acceleration cards, which can be installed in any place, including remote servers with untrusted Operating Systems and system administrators, while still providing privacy-preserving guarantees;
- ultimately, enable federated architectures based on accelerated computing services, which will promote open, privacy-aware data sharing and processing.

As such, the proposed solution will allow moving from centralized settings towards fully privacy-aware federated architectures, promoting secure data exchange, data ownership and access right tracking, along with high-performance computing services.

3.1 Prototype

The project will release a fully operational environment for privacy-sensitive accelerated HPC/big data application, also relying on the outcome of previous research projects, particularly a H2020 project in the area of High-Performance Computing (H2020-FETHPC MANGO project ID 671668, where one of the authors of this deliverable served as part of the core research group and leader of WP3 dealing with compute node technologies), then extended in the H2020-FETHPC RECIPE project ID 801137 with high-bandwidth interconnect support. The outcome from previous projects will include access to cutting-edge high-performance computing technologies and guarantee the technical feasibility of the ambitious research plan described here. In particular, the prototypical platform, shown below, features multiple datacenter-grade FPGA acceleration cards (namely, Xilinx U280 and U250 Alveo cards), hosted by a dedicated PCIe backplane enabling peer-to-peer data communication, directly connected to an InfiniBand network adapter based on 200Gb/s HDR communication standard. An illustrative representation is depicted in the figure below.

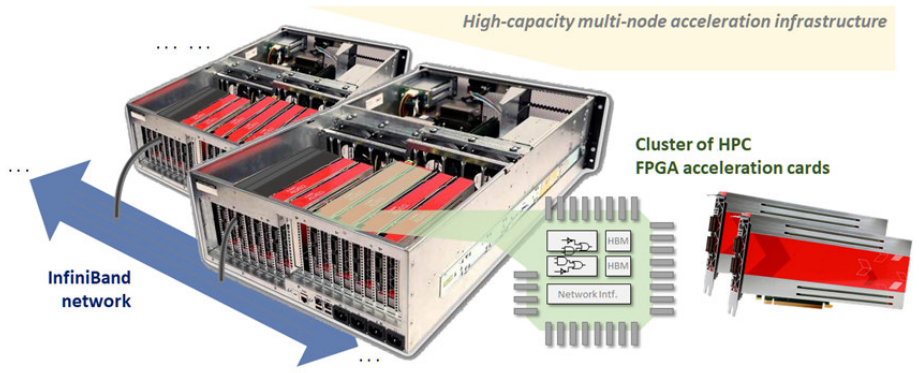

Figure 2: High-level view of the prototypical TEE-enabled accelerated system explored in Flagship 4.

This setup will be representative of cutting-edge FPGA-based compute technologies, but also ultra high-bandwidth inter-node interconnects, which will be key for HPC applications where often multi-terabyte datasets need to be transferred and accessed for data analysis. Hence, augmented with privacy-preserving acceleration support, this prototype will serve as an excellent showcase for the solution explored here, suitable for two types of deployment in real settings:

- on-premise machines, where researchers or institutions directly acquire and own privacy-enabled acceleration nodes for deployment in local infrastructures, and
- cloud-based access, fully compliant with the federated approach envisioned by the project and inherently relying on its privacy-preserving trusted acceleration environment.

4 Secure virtualization

Hardware-assisted virtualization takes advantage of CPU virtualization technologies (e.g., Intel VT-x [29], AMD-V [2], ARM VHE [12]) to implement *full virtualization*, i.e., emulating a complete machine to run unmodified guests. These hardware extensions introduce a novel (and higher) level of privilege for the hypervisor. This way, developers can implement *virtual CPU (vCPU)* abstractions that can run a guest OS at a lower level of privilege compared to the hypervisor. If the guest OS needs to execute a *sensitive* instruction (e.g., page table update, interrupt handling, etc.), the execution traps, and the control is passed to the hypervisor. The switch from the guest OS to the hypervisor is called *VM exit*. A VM exit also involves a change in the privilege level, from the VM to the hypervisor's most privileged mode. This passage, however, can put isolation at risk [32, 31, 30], possibly leading to hypervisor crashes (together with all running VMs) or denial of service issues (hangs, low responsiveness, etc.). Uncovering such *isolation issues* is a major concern and a compelling open challenge to foster the adoption of hypervisors in critical industrial domains. This is especially true for hardware-assisted virtualization, as the search for isolation issues should include all the possible VM states reached by an extremely high number of combinations of CPU instructions.

In hardware-assisted virtualization architectures, the hypervisor code is mainly run when *VM exits* occur. Virtualization bugs can be discovered by hardening such parts of the hypervisor source code. Identified gaps (see *Deliverable 4: Survey of state-of-the-art approaches and gap analysis*, Section 4.6)) include a non-negligible effort to apply security assessment techniques to hardware-assisted virtualization solutions.

For example, existing hypervisor fuzzing solutions fall short when targeting hardware-assisted virtualization, since they *i)* mostly target I/O virtualization [39, 28, 18, 33, 7], starting from the same VM state, thus leaving behind the vCPU abstraction, which is the core of hardware-assisted hypervisors; *ii)* face serious issues related to seed generation (build valid VM states to start the fuzzing [14, 43], generation of valid seeds to be corrupted to accelerate the fuzzing, which may require the use of an ad-hoc OS within VM [39, 40] or construct seeds [3, 43, 14, 15]); *iii)* require a deep knowledge of the underlying hardware (i.e., the knowledge of an operating system), trying to reach valid VM states from a dumb sequence of inputs, which lead to incurring in several crashes of the test VM [45], which then requires resetting the test, with a non-negligible impact on the testing time efficiency.

To overcome the above limitations, including Gap05 identified in D4.FL4, in Flagship 4 we will address these limitations by proposing a new framework to *record* (learn) sequences of inputs (i.e., VM seeds) from the real guest execution (e.g., OS boot), *replay* them as-is to reach valid and complex VM states. Such a record and replay mechanism can be used in different security assessment techniques for hardware-assisted virtualization, such as fuzzing (use the replay mechanism to submit mutated valid VM seeds) and runtime verification/monitoring (detect hypervisor misbehavior according to recorded normal behaviors).

4.1 Proposed architecture

The record and replay framework we propose allows to easily *record* (learn) complex VM scenarios, by executing guests; further, the *replay* mechanism allows submitting recorded VM behaviors as a train of recorded VM exits, with no need to execute guest workloads. The main challenges to be addressed are in the following:

1. **Automatic VM seed generation**: reduce the manual effort in VM seed generation, to minimize the required knowledge to develop security assessment techniques such as fuzzing and runtime verification/monitoring for hardware-assisted hypervisors;

2. **Accurate VM seeds generation**: generate VM seeds that are accurate in reproducing the real VM behaviors;

3. **Efficient submission of VM seeds**: submit VM seeds in a lightweight fashion, without requiring specific VM guest workload to be executed.

Figure 3 depicts a sketch of the high-level architecture of the proposal. The ***VM behavior*** is a sequence $VM\_exit\_trace = \{VMexit_1, ..., VMexit_N\}$ that is the flow of *VM exits* triggered by a workload to reach a valid VM state. The ***VM seed*** includes the pairs of VMCS {field, value} read via `VMREAD` instructions, and the values of general-purpose registers (GPR), both obtained during the handling of a VM exit within the *VM_exit_trace*. The **recording** component aims to collect a set of information observed while executing the VM, i.e., the *VM behavior*. The **replaying** component allows submitting recorded or manually built *VM seeds* to the hypervisor. The **manager**, exposes an interface that can be used by a *user-space application (CLI)* to *(i)* choosing between operation modes, i.e., *record* and *replay*; *(ii)* retrieve *VM seeds* and metrics during the *record mode*; *iii)* submitting *VM seeds* during the *replay mode*. Such a solution will jointly address Gap06, Gap07, and Gap08.

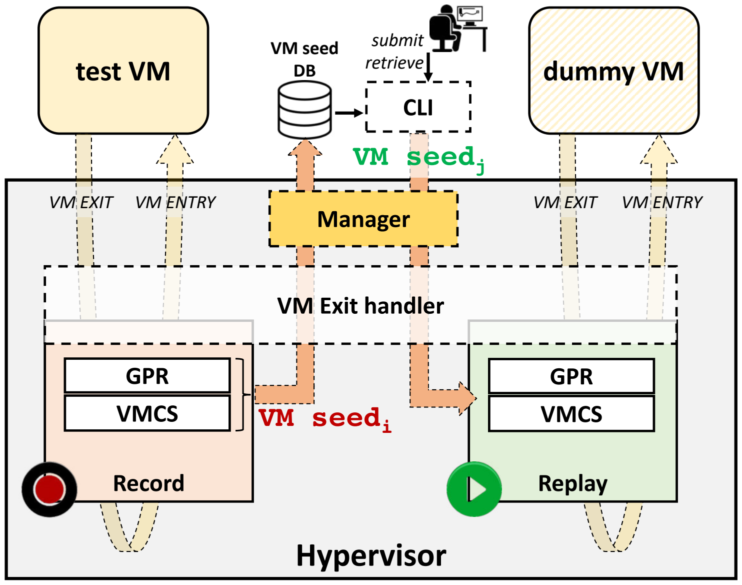

Figure 3: Overview of framework design

4.2 MPSoCs virtualization

The use of modern MPSoCs to build virtualized cyber-physical systems with differentiated criticality levels on the same board is not an easy task. The configuration phase for each technological stack as well as the isolation and the interconnection between the virtual environments requires a strong effort and specialized expertise. This is especially true when the system must be integrated into critical environments such as the automotive, railway, or avionics where safety, security and real-time requirements have to be fulfilled. In these circumstances, over-provisioning is frequently the solution: each board is used to run a single application. Nevertheless, this approach has the disadvantage of preventing migration, integration of new features, and code portability limiting non-functional requirements necessary for the fog computing paradigm such as scalability, agility, reconfigurability, and workload consolidation, as well as obviously increasing the total cost for the implementation of the system.

To overcome these limitations, we propose a framework that extends the traditional hypervisor concept: the *Omnivisor*. It provides the ability to run virtual environments in isolation with different requirements, from strict temporal predictability to high performance, making full use of existing asymmetric processors, accelerators, and re-programmable hardware (see Fig.4). Each VM can run on different processors and can share a resource or take full ownership of it according to its criticality and needs. Code with hard real-time requirements can run over RPUs while code that needs higher performance can run over APUs using accelerators such as the GPU. This is feasible only if the Omnivisor has full control of the resources and implements the security mechanism that can guarantee temporal and spatial isolation of the virtual environments. The implementation of the Ominvisor concept can benefit from the recent advances on CPU, memory, and I/O virtualization. However, despite the availability of these technological advancements, there is still a lack of solutions to virtualize the asymmetric processors on board, namely the RPU virtualization.

To realize a dependable solution for RPU's virtualization, the following requirements need to realized:

- *Multi-Tenancy Abstraction*: The hypervisor must abstract the RPUs to more than one VM, making the latter believe to be the only one using it.
- *Deterministic Communication Technologies*: The current Asymmetric Multi-Processing (AMP) communication technologies (adopted for inter-processor communication in MPSoCs) must be improved to realize deterministic and secure applications with strict temporal constraints.
- *Isolation Methodologies*: The hypervisor have to manage parallel communication channels ensuring isolation for security and assure a certain amount of communication bandwidth to each channel (useful for timing analysis and incremental development and certification).

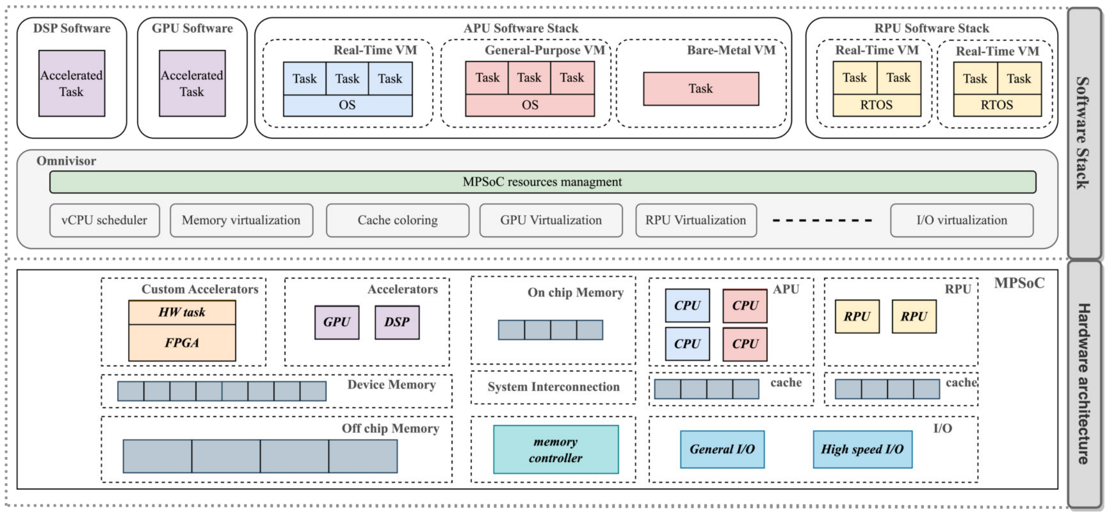

Figure 4: Omnivisor Design.

5 Federated Learning

5.1 Federated Learning without Gradient Descent

Most FL algorithms are currently based on some form of gradient descent optimization of the model parameters. While this approach is well suited for the FL setting, it makes it difficult to develop systems where traditional ML algorithms (e.g., SVMs, decision trees, K-NN, etc.) would be preferred over neural networks and other gradient-based approaches. A first attempt to overcome this problem has been presented in [36], where techniques based on distributed versions of AdaBoost have been tailored to the FL setting.

While the approach looks promising, the algorithms have been developed under the assumption of being in a horizontal and cross-silo FL setting. The authors, in fact, assume that the number of entities participating in the federation is low, that they are always available, and that they are well connected. Also, the proposed algorithms cannot work in a vertical FL setting.

The resulting algorithms make heavy usage of communication resources and cannot scale to settings where the number of clients is high. These problems do not appear, however, unsurmountable. To address the gaps identified in D4.FL4, particularly Gap09, it would be very interesting to extend these algorithms to cope with cases that are more challenging from the point of view of the client availability and of the constraints on the communication costs.

5.2 Secure Federated Learning

In the traditional FL setting, data cannot be moved from their original location due to privacy concerns. This requirement excludes shared infrastructures for model training, such as public cloud and HPC facilities. However, these architectures' computing power and communication speed are fundamental for training modern large-scale deep networks [6, 37]. Trusted execution environments are a potential solution to this issue. As long as privacy and integrity are preserved throughout the training process, nothing prevents data from being moved to remote locations.

Implementing end-to-end secure FL processes is non-trivial, as it requires a careful application of the security-by-design principle to all the steps of the learning pipeline, such that unencrypted data should never leave the perimeter of the secure enclave. Moreover, the aggregation plane should rely on resistant aggregation functions to avoid attacks, such as model poisoning and byzantine agreements [24]. Besides increasing implementation complexity, these techniques add overhead to a training process's computation and communication phases, hindering performance.

The first research step is to measure such overhead, comparing it with the benefits of high-end hardware technologies provided by cloud and HPC systems. The second step is to find generic methodologies to predict the extent of the overhead starting from a coarse-grain characterization of the training process, e.g., the model size, the amount of data to protect, and the difference in computing power between local and remote resources.

The natural culmination of this process consists of developing a configurable FL framework that can guide users in tweaking the security vs performance trade-off according to the level of trust of different portions of the federated infrastructure and the time constraints for the overall training process.

5.3 Resisting to privacy attacks

In addition to the above considerations, one would also need to cope with possible privacy attacks by entities participating in the federation. Model inversion attacks, for instance, exploit the exchange of gradient information between rounds of the FedSGD algorithm to reconstruct examples from other clients' datasets. A number of techniques can be deployed to overcome these problems: secure multiparty computation, differential privacy, homomorphic encryption, or methods that do not share the entire model (e.g., split learning) appear to be good candidates for counteracting these attacks. Unfortunately, all these methods have problems. Multi-party computation requires additional communication. Differential privacy risks to affect the federated model performances too heavily. Homomorphic encryption is computationally very demanding. Not sharing the entire model is less studied, but split learning-based methods are very costly from the communication point of view. A candidate solution to address Gap10 and Gap11 may try to share with the aggregator only the layers nearer to the output and some technique to force the embedding layers (kept local to the clients) to converge to a common solution without requiring to share any data.

6 Trustworthy AI

6.1 Assessing and ensuring AI trustworthiness: explainability, robustness, and fairness

The Big data availability and the widespread high-performance computing (HPC) have promoted the development of machine learning (ML) and artificial intelligence (AI) models. However, as AI models become increasingly pervasive, assessing and enforcing their trustworthiness and reliability is critical. Trustworthy AI research aims to develop AI systems that are explainable, robust, and fair, making AI trustworthy [23]. Despite the recent progress, several gaps still need to be addressed to ensure trustworthy AI models. We envision contributing to this aim via the following efforts.

Benchmarking explainability. Explainable AI (XAI) indicates the set of techniques to enable human users to understand the outcomes of ML models. Explainability is a crucial requirement of trustworthy AI. Therefore, multiple research efforts have proposed XAI explanation methods for model understanding. However, there is still a lack of easy access to using and comparing XAI explanation methods. There is a need to assess the quality of explanations generated by XAI. Specifically, we should compare explanations with respect to their adherence to the model behavior and human reasoning. To effectively address Gap12, we are working on an open-source library called ferret. ferret is a framework that simplifies the use and comparisons of XAI methods. We proposed it in the context of Natural Language Processing (NLP). Benchmarking XAI approaches is crucial in high-risk and impactful applications such as automated legal document processing and hate speech detection. We will benchmark and evaluate XAI approaches in such critical applications, proposing ad-hoc solutions for each domain.

Trustworthy AI in multiple application domains and for multiple data types. Big Data and HPC applications have enabled the adoption of ML models in critical fields such as ranking systems and law. Ensuring these systems are reliable, ethical, and aligned with societal values is crucial. We envision assessing and ensuring AI trustworthiness in critical application domains, proposing ad-hoc solutions depending on the application context. Moreover, ML applications consider various types of data, from structured to unstructured data as images, text, and speech. However, existing trustworthy AI techniques are often designed to address specific data types. Hence, as a strategy to address Gap13 and Gap14, we plan to generalize trustworthy methods for various data types and application domains. Among the existing techniques, we first target to address the problem of identifying data subgroups with anomalous behavior in the data [8, 35, 34]. Identifying these subgroups plays an important role in assessing three key requirements of trustworthy AI: fairness, robustness, and explainability. DivExplorer [34] is an anomalous subgroup identification technique we recently proposed. The proposed algorithm is based on effectively integrating frequent pattern mining algorithms to identify subgroups with peculiar behavior. Moreover, the approach leverages game-theory notions to allow for the understanding of the model behavior among subgroups. However, it is specifically designed only for structured data and extensively applied for classification tasks. We will work on novel methodologies to enable the study of the behavior for multiple tasks, application domains, and different data types as speech data.

6.2 Models and tools for privacy assessment in internet data

Users' data has fuelled the growth of the current economic ecosystem in the Internet. Online advertising and marketing have driven developments in this space, transforming a decades-old industry and creating some of the biggest businesses (and in a few cases, controversies) of our time. In fact, the online advertising industry is breaking records year after year, both in terms of growth and overall value.

In some aspects, this economy is primitive: the source of value – or raw material – are the users, and they have no choice but to give away their goods (data) to a very few companies against whom they have no bargaining power. In exchange for their goods, users receive a range of services, some of which are now essential to everyone's digital life: web search, connecting with other people, shopping, etc.

This system creates tension between the utility of data and the privacy of those users to which such data belong. We aim to propose new solutions to find a more balanced trade-off in which the privacy needs of the users are not subordinate to their data's economic value, but rather constitute a fundamental line not to cross.

To design better solutions for privacy-preserving advertisements on the Web, we cannot overlook a better understanding of this fragmented and ever-evolving field. Hence, to address Gap15, Gap16, Gap17, and Gap18, we aim to closely scrutinise solutions proposed by the industry, which have to be discussed in a dialectic approach with privacy-advocating and academic parties, in particular focusing on the study and understanding of the new proposals like the FLoC and the Topics API that are designed to balance the end-user privacy and the data utility for the online advertising companies. We will propose models to easily gauge the impact of the privacy-preserving approaches and validate their design with actual data collected from end-users.

7 Social media data analysis

Social media analysis has advanced significantly in recent years, with a range of techniques and tools available for data collection and analysis from various social media platforms. Overall, the ability to collect and analyze social media data at a large scale provides valuable insights across various aspects of society. However, we identified some potential research gaps and open questions in scalable social media data collection and analysis. In summary: quality control, data privacy and ethical concerns, machine learning and AI, cross-platform analysis and real/time analysis (Gap19, Gap20, Gap21, Gap22, Gap23).

To address the above listed gaps, we aim to closely investigate part of the previous questions and answer them, focusing on the important scalability issues and their relation with HPC and Big Data infrastructure and software for data collection, storage, and processing. Some of the techniques that will be involved will relate to graph data modelling and analysis, followed by recent ML algorithm designed for networks, such as GNN and temporal GNN. Particular effort will be devoted to the study of different sources of information, including instant messaging platform, such as Telegram groups. Finally, the developed techniques will help to shed light on these complex ecosystems, discover interesting patterns (e.g., polarization) and highlight possible misuse, from cybersecurity (e.g., phishing) to misinformation.

8 Numerical Analysis

8.1 A fast, accurate and stable numerical open-source software for approximation using hyperparameters tuning

Several parameters have to be chosen in meshfree approximation: the shape parameter in the radial kernel, the number of patches and the radius of the subdomains in a local approach. An application of approximation to the interpolation and classification of signals on graphs, has been considered and also in this context the determination of optimal parameters plays a fundamental role. From a deterministic point of view, the target is to exploit cross validation and likelihood estimation techniques in combination with strategies of univariate global optimization with pessimistic or optimistic improvements and to design fast and stable algorithms. Moreover, to address Gap24, we will assume also a statistical approach which can be useful also in Machine Learning and Big Data, for example considering Bayesian Optimization of the hyperparameters. In particular, Bayesian Optimisation can be used to simultaneously search the optimal values of the shape parameter and the radius in RBF interpolation together with the Partition of Unity Method. Also in this case we need to implement efficient algorithms. The algorithms will be translated in Matlab/Python so creating a fast, accurate and stable numerical software package useful in applications to Machine Learning and Big Data.

8.2 A software package for interpolation and classification on large graphs and networks

Communities detection is an edge-cutting topic in graph theory, indeed different approaches may be used. In that respect, to address Gap25, we have in mind to construct a method with a divisive technique where the centrality of the interpolation nodes is the main criterion for splitting. In particular, we would use a procedure allowing overlapping among communities. Our iterative algorithm will repeat the following steps until some conditions on the presence of interpolation nodes and modularity are satisfied. In detail, in the splitting process we find the two interpolation nodes with the highest Katz centrality in each community, if they exist, and divide the community into two sub-components, each one containing one of the samples. When the splitting process terminates, the small communities containing less than 2% of vertices are merged with the most similar big community, according to the Jaccard index. Finally, the overlapping expansion is performed for each community, where the more edges a community shares with the others, the more it is augmented. This target will be finalized to the production of open-source software in Matlab and Python usable by the scientific community.

8.3 Parallel software for data approximation and signal processing on graphs

Since the idea is to use the algorithms in Machine Learning and Big Data applications, a parallel implementation will be obtained. All the numerical experiments and tests will be carried out on the infrastructure for high performance computing MathHPC, virtual cloud server of the structure HPC4AI (High-Performance Computing for Artificial Intelligence) at the University of Torino.

9 Trusted Distributed Workflows

A zero-trust approach to distributed workflow orchestration would be a game changer for a broad class of research fields, enabling end-to-end trusted experiments at the HPC scale. Secure enclaves and their remote attestation mechanisms can guarantee the privacy and integrity of remote computations. The library operating system (libOS) abstraction allows running unmodified applications under complete enclave protection.

Configuring security and privacy aspects of a distributed workflow execution at the coordination level is crucial for portability. Currently, libOSes enclave interfaces like Gramine [42] and Occlum [41] require users to configure declarative manifest files with enclave parameters, e.g., memory size or the maximum number of threads, and a list of trusted files that should be checked for integrity and executed.

This approach has two main drawbacks. First, the manifest format varies from one libOS to another, hindering portability and reproducibility. Second, properly configuring a libOS manifest is non-trivial for domain experts, as it requires a deep knowledge of the libOS implementation and the enclave architecture. Even worse, both security and performance tend to be very sensitive to misconfigurations.

Integrating the capability to express high-level security requirements directly in a vendor-agnostic workflow coordination language can solve both issues, moving the responsibility of generating libOS manifests with proper configurations to the workflow management system. However, to properly address Gap26 and

Gap27, it is necessary to devise a general methodology to infer manifest parameters from the workflow description and the target execution environment, which is a challenging task.

The StreamFlow workflow management system [9] is a perfect laboratory to experiment with libOSes integration for secure distributed workflows. It can orchestrate hybrid workflows, where each step can be mapped onto a different execution location (e.g., a cloud environment or an HPC facility). Such locations can be heterogeneous and independent of each other and may or may not share a single file system. In the latter case, StreamFlow automatically manages data movements. Plus, several aspects of workflow orchestration, such as scheduling, data transfers, fault tolerance, and task offloading, can be extended using the StreamFlow plugin system.

StreamFlow relies on the Common Workflow Language (CWL) open standard [11], a vendor-agnostic declarative coordination language based on the data flow programming model. An existing CWL extension allows marking a set of inputs as `Secrets`, denoting sensitive data. If a step has at least one input marked as secret, StreamFlow automatically wraps it into a Gramine libOS and executes it using the Intel® SGX enclave.

At runtime, remote attestation is used to exchange encryption keys through an end-to-end secure channel between the StreamFlow control plane and the encrypted memory of the enclave. Input data are encrypted, moved to the remote location, and decrypted directly inside the enclave. Output data are encrypted in the enclave before being transferred back to the user.

References

[1] Moreno Ambrosin, Mauro Conti, Riccardo Lazzeretti, Md Masoom Rabbani, and Silvio Ranise. Collective remote attestation at the Internet of Things scale: State-of-the-art and future challenges. *IEEE Communications Surveys & Tutorials*, 22(4):2447–2461, 2020.

[2] AMD Technology. AMD64 Architecture Programmer's Manual Volume 2: System Programming.

[3] Nadav Amit, Dan Tsafrir, Assaf Schuster, Ahmad Ayoub, and Eran Shlomo. Virtual CPU validation. *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[4] Nadarajah Asokan, Ferdinand Brasser, Ahmad Ibrahim, Ahmad-Reza Sadeghi, Matthias Schunter, Gene Tsudik, and Christian Wachsmann. Seda: Scalable embedded device attestation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 964–975, 2015.

[5] Richard Barry et al. Freertos. *Internet, Oct*, 2008.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[7] Alexander Bulekov, Bandan Das, Stefan Hajnoczi, and Manuel Egele. Morphuzz: Bending (Input) Space to Fuzz Virtual Devices. In *USENIX Security Symposium*, 2022.

[8] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Slice finder: Automated data slicing for model validation. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1550–1553. IEEE, 2019.

[9] Iacopo Colonnelli, Barbara Cantalupo, Ivan Merelli, and Marco Aldinucci. StreamFlow: cross-breeding cloud with HPC. *IEEE Transactions on Emerging Topics in Computing*, 9(4):1723–1737, 2021.

[10] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 857–874, 2016.

[11] Michael R. Crusoe, Sanne Abeln, Alexandru Iosup, Peter Amstutz, John Chilton, Nebojsa Tijanic, Hervé Ménager, Stian Soiland-Reyes, Bogdan Gavrilovic, Carole A. Goble, and The Cwl Community. Methods included: Standardizing computational reuse and portability with the common workflow language. *Communications of the ACM*, 65(6):54–63, 2022.

[12] C. Dall, S. Li, J. T. Lim, J. Nieh, and G. Koloventzos. ARM virtualization: performance and architectural implications. In *Annual International Symposium on Computer Architecture*, pages 304–316. IEEE, 2016.

[13] Karim Eldefrawy, Gene Tsudik, Aurélien Francillon, and Daniele Perito. Smart: secure and minimal architecture for (establishing dynamic) root of trust. In *Ndss*, volume 12, pages 1–15, 2012.

[14] Pedro Fonseca, Xi Wang, and Arvind Krishnamurthy. MultiNyx: a multilevel abstraction framework for systematic analysis of hypervisors. *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[15] Xinyang Ge, Ben Niu, Robert Brotzman, Yaohui Chen, HyungSeok Han, Patrice Godefroid, and Weidong Cui. HyperFuzzer: An Efficient Hybrid Fuzzer for Virtual CPUs. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[16] Daniel Genkin, Itamar Pipman, and Eran Tromer. Get your hands off my laptop: Physical side-channel key-extraction attacks on pcs. *Journal of Cryptographic Engineering*, 5(2):95–112, 2015.

[17] Prasanna Hambarde, Rachit Varma, and Shivani Jha. The survey of real time operating system: Rtos. In *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, pages 34–39. IEEE, 2014.

[18] Andrew Henderson, Heng Yin, Guang yao Jin, Hao Han, and Hongmei Deng. VDF: Targeted Evolutionary Fuzz Testing of Virtual Devices. In *RAID*, 2017.

[19] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[20] Patrick Koeberl, Steffen Schulz, Ahmad-Reza Sadeghi, and Vijay Varadharajan. TrustLite: A security architecture for tiny embedded devices. In *Proceedings of the Ninth European Conference on Computer Systems*, pages 1–14, 2014.

[21] Nick Kossifidis, Joe Xie, Bill Huffman, Allen Baum, Greg Favor, Tariq Kurd, and Fumio Arakawa. Epmp: PMP Enhancements for memory access and execution prevention on Machine mode (Smepmp). 2021.

[22] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[23] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Comput. Surv.*, 55(9), jan 2023.

[24] Pengrui Liu, Xiangrui Xu, and Wei Wang. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecur.*, 5(1):4, 2022.

[25] LOWRisc. Ibex: An embedded 32 bit RISC-V CPU core. 2021.

[26] Enno Lubbers and Marco Platzner. Reconos: An rtos supporting hard-and software threads. In *2007 International Conference on Field Programmable Logic and Applications*, pages 441–446. IEEE, 2007.

[27] Jun Ma, Ting Chong, Lei Li, Hao Liu, Xige Zhang, Liang Liu, and Yidong Yuan. Construction of RISC-V Lightweight Trusted Execution Environment Based on Hardware Extension. In *2021 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, pages 237–242. IEEE, 2021.

[28] Cheolwoo Myung, Gwangmu Lee, and Byoungyoung Lee. MundoFuzz: Hypervisor Fuzzing with Statistical Coverage Testing and Grammar Inference. In *USENIX Security Symposium*, 2022.

[29] Gil Neiger, Amy Santoni, Felix Leung, Dion Rodgers, and Rich Uhlig. Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. *Intel Technology Journal*, 10(3), 2006.

[30] NIST. CVE-2010-0435.

[31] NIST. CVE-2011-1936.

[32] NIST. CVE-2020-2732.

[33] Gaoning Pan, Xingwei Lin, Xuhong Zhang, Yongkang Jia, Shouling Ji, Chunming Wu, Xinlei Ying, Jiashui Wang, and Yanjun Wu. V-Shuttle: Scalable and Semantics-Aware Hypervisor Virtual Device Fuzzing. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[34] Eliana Pastor, Luca De Alfaro, and Elena Baralis. Looking for trouble: Analyzing classifier behavior via pattern divergence. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1400–1412, 2021.

[35] Eliana Pastor, Andrew Gavgavian, Elena Baralis, and Luca de Alfaro. How divergent is your data? *Proceedings of the VLDB Endowment*, 14(12):2835–2838, 2021.

[36] Mirko Polato, Roberto Esposito, and Marco Aldinucci. Boosting the federation: Cross-silo federated learning without gradient descent. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10. IEEE, 2022.

[37] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 8821–8831, 2021.

[38] Debapriya Basu Roy, Manaar Alam, Sarani Bhattacharya, Vidya Govindan, Francesco Regazzoni, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. Customized instructions for protection against memory integrity attacks. *IEEE Embedded Systems Letters*, 10(3):91–94, 2018.

[39] Sergej Schumilo, Cornelius Aschermann, Ali Reza Abbasi, Simon Wörner, and Thorsten Holz. HYPER-CUBE: High-Dimensional Hypervisor Fuzzing. In *NDSS*, 2020.

[40] Sergej Schumilo, Cornelius Aschermann, Ali Reza Abbasi, Simon Wörner, and Thorsten Holz. Nyx: Greybox Hypervisor Fuzzing using Fast Snapshots and Affine Types. In *USENIX Security Symposium*, 2021.

[41] Youren Shen, Hongliang Tian, Yu Chen, Kang Chen, Runji Wang, Yi Xu, Yubin Xia, and Shoumeng Yan. Occlum: Secure and efficient multitasking inside a single enclave of intel SGX. In James R. Larus, Luis Ceze, and Karin Strauss, editors, *ASPLOS '20: Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, March 16-20, 2020*, pages 955–970. ACM, 2020.

[42] Chia-che Tsai, Donald E. Porter, and Mona Vij. Graphene-sgx: A practical library OS for unmodified applications on SGX. In Dilma Da Silva and Bryan Ford, editors, *2017 USENIX Annual Technical Conference, USENIX ATC 2017, Santa Clara, CA, USA, July 12-14, 2017*, pages 645–658. USENIX Association, 2017.

[43] Qiuchen Yan and Stephen McCamant. Fast PokeEMU: Scaling Generated Instruction Tests Using Aggregation and State Chaining. *Proceedings of the 14th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, 2018.

[44] Yuanhai Zhang, Hui Han, Jinxing Jiao, Guizhou Xu, Gang Chen, and Kai Huang. Brief industry paper: Sylixos: A secure and compatible rtos with constant scheduling on smp. In *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 449–452. IEEE, 2021.

[45] Xiaogang Zhu, Sheng Wen, Seyit Camtepe, and Yang Xiang. Fuzzing: a survey for roadmap. *ACM Computing Surveys (CSUR)*, 54(11s):1–36, 2022.