Thesis Summary:

University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers
Computer Science and Engineering Department

PHD THESIS

Privacy in Critical Infrastructure

Scientific Adviser:
Prof. PhD. Eng. Răzvan-Victor Rughiniș

Author:
Eng. Ioan-Mihail Stan

Bucharest, 2022

Abstract

With the development of the spectrum of online services and the massive migration of the masses, voluntary or forced by major global contexts, towards a digital interaction, the privacy perspective on data has a new dimension. From minor analysis for commercial purposes to manipulation in order to destabilize communities, data has become a form of exercising power. Thus, awareness of aspects related to protection against exposure in the online environment is mandatory.

The concept of privacy implies the preservation of the personal environment and the exercise of the right not to be invaded by unauthorized persons. In the online environment, setting up privacy boundaries is a way to manage personal, private data with the goal not to be exposed to third parties without a prior, deliberate agreement. Furthermore, to better emphasize the issue from a technical perspective, it is essential to express the notion of privacy in a complex context, susceptible to being targeted, by malicious entities, due to its relevance and scale. Thus, discussing data protection issues in critical infrastructures, it becomes relevant. Critical infrastructure are those infrastructures considered essential by a state or form of government for the optimal management of citizen life. In general, from the perspective of state security, the first industries or sectors that become targets, in the event of a war or a massive cyber-attack, are the critical systems.

This thesis outlines the understanding of privacy aspects from five points of observation. The first is represented by the accessibility of data as a form of presentation and also as a form of delivery. Aspects such as Software Development Life Cycle and Automation are being studied in order to establish quick reaction mechanisms to vulnerabilities or exploits. The second is the anonymity of data, exploring methods and methodologies of segregating the real identity from the one in digital space. The methods of hiding the identity from blockchain technologies are analyzed and the existing models are supplemented with intermediate stages. The structure of the scientific discussion revolves around the development of a national voting system based on permissioned blockchain solutions. The third is related to data transmission where blockchain technologies are considered for their intrinsic ability to provide communication topologies. The qualities and problematic aspects in the communication function are analyzed and compared with the cloud computing paradigm. The fourth is related to the distribution and orchestration of data in which the feasibility of using containerized infrastructures is validated. The isolation taxonomy is presented and the scheduling methods for containerized data services are analyzed. One can see how user policies can have a better representation with less complex isolation taxonomies. Furthermore, the study proposes a mix of distinct governance models for distributed infrastructures (High Performance Computing, Grid and Cloud Computing) in order to bring together the qualitative aspects of each, including also the privacy dimension. The fifth shows the importance of establishing a risk assessment for data services. The relevance of using honeypot infrastructures for the exposure of developed assets is emphasized. Therefore, a development framework for such solutions has been provided and also a consistent cloud deployment methodology facilitating coexistence with legitimate production.

Keywords: Security, Privacy, Data Protection, Blockchain, HPC, Grid, Containers, Cloud, Honeypots, Automation

Context

In recent years, the global community has experienced several crises and major prospective developments that have led to an increased acceleration of digital transformation. The Covid-19 crisis showed us how we can cross the boundaries of physical distance, moving the entire activity online. There has been a major increase in the use of co-working services, video conferencing tools, VPN

solutions [1] and so on, as a consequence of major reforms in sectors such as education, information technology or government services. During the lockdown, the behavioral habits of the users have also changed. They supplemented part of the recreational activity in the online environment. Major increases in online commerce could be observed, which subsequently led to a crisis in the supply chain generating significant delays. At the same time, the semiconductor crisis and chip shortage made critical industries unable to deliver on time, affecting important sectors such as critical manufactures, transportation and other sectors relying on electronic systems. At the same time, the post-pandemic tensions produced conflicts and wars that destabilized the financial and energy sectors, especially in the European space through the conflict between Russia and Ukraine. Last but not least, voted in 2016 and implemented in 2018, the General Data Protection Regulation, set those companies that handle personal data of the European Union citizens, in a continuous restructuring of the IT infrastructure in order to cover the new critical provisions. As can be seen, such major situations imply additional stress and generate the premises of an endurance test in some of the key industrial sectors.

With the massive migration of daily activity online, a significant increment in network attacks and user exposure, in the online space, could also be observed [2]. Thus, the problem of security and especially privacy has reached new heights, observing a rise in attack methods, exploitable but still intensively used applications and success rates in compromising public assets. It becomes more and more relevant to identify and understand, in detail, exploitation opportunities and the countermeasures. All defense mechanisms can rely on general methodologies and industrial standards or can be customized for each type of the exposed assets. It becomes relevant to talk about the use of honeypots on a large scale for creating security reports per exposed service, about intelligent orchestration of applications that also cover the need to preserve privacy, about anonymous communication and secure transfer of digital assets over technologies such as blockchain, about sharing large infrastructures between multiple tenants with the aim to facilitate digital transformation, and so on.

The thesis covers multiple aspects of privacy preservation in various infrastructures. It comprises both methods and methodologies, proposes solutions, analyzes critical situations, observes the opportunities for digital transformation, suggesting forms through which the transition can be made with the minimum effort required to cover relevant security and privacy needs for the management of critical data, and also other relevant aspects. The focal point is set on critical infrastructures, those infrastructures that are expensive, sensitive to global or local crises and which, in the context of the inability to deliver services, affect the masses. Digital critical infrastructures tend to implement the multi-tenancy paradigm and are mostly based on the distribution of computational effort up to decentralization. Thus, the scientific analysis will primarily isolate and consider the following attributes in regard to defining critical infrastructure: massive deployments, distributed, centralized or decentralized infrastructures, geographically dispersed or maintained in large data centers, being able to serve multiple tenants.

Motivation and Objectives

Preservation of data privacy is a real problem with global impact. Understanding the opportunities and methods by which data protection can be guaranteed in small, medium and large infrastructures is a relevant concern in the activity of IT engineers, as they are often put in a position to react quickly to any disruptive event and to identify potential break-in points. Thus, the current thesis analyzes the global context from the perspective of complex and expensive infrastructures which must ensure an adequate level of data privacy and security. The topics studied cover a wide range of points of interest around data confidentiality preservation:

- light virtualization with containers for isolating the critical data processing context
- orchestration of large, containerized infrastructures to ensure the necessary data protection and consistency from the deployment stage
- blockchain for its intake in secured and anonymous communication and also for the storage function suitable for critical data
- honeypots as a way to establish organic parameters for measuring the security and privacy level of exposed services
- large distributed systems such as grid computing or high-performance computing, centralized or decentralized, and the policies they apply to maintain security, confidentiality, and consistency of data
- automation to ensure a quick reaction to unexpected events, exploits or cyber attacks
- methods and methodologies for rapid digital transformation without losing sight of the security and confidentiality of the managed data

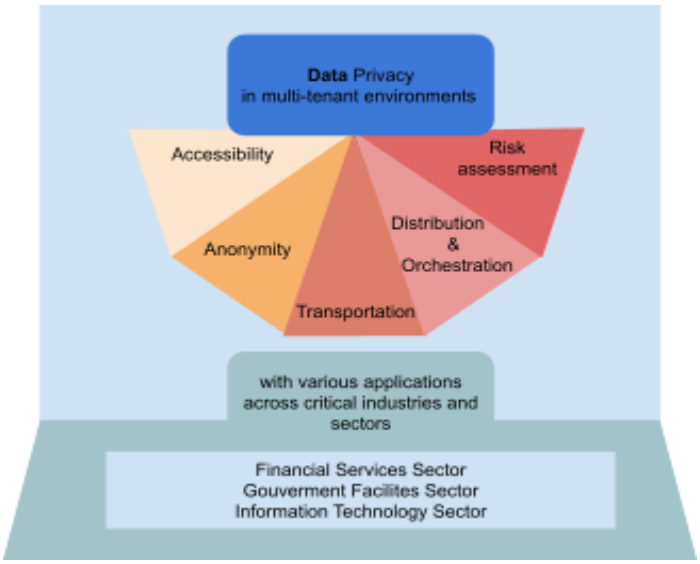


Figure 1 - Thesis Objectives and case studies

The objectives of the thesis are illustrated in Figure 1, proposing five dimensions through which data privacy assurance is analyzed. Those infrastructures under observation are critical infrastructures, proposing three formal examples of industrial sectors on which I have built case studies. However, the thesis addresses a wider scope of applicability, the studies generated being also valid and portable to non-critical infrastructures of variable sizes. The five dimensions proposed are:

- Accessibility, which defines methods and methodologies for exposing data in order to be consumed, proposes automation of development and release cycles, proposes methods of presentation of data services in shared infrastructures.
- Anonymity, which highlights forms of establishing the separation of real and digital identities with no need to obstruct participation in essential actions such as civic or governmental processes, where consistent authentication is required.
- Transportation, which presents methods of maintaining confidentiality and consistency of data while moving or trading private information between end-nodes, exposes limitations of communication facilitators.
- Distribution and Orchestration, which presents mechanisms that perform data isolation and packaging as part of the effort to ensure data secrecy and portability, methods to distribute or

localize data, methods to schedule and spin-up data processors near the data stores in order to avoid any possible delays generated by data migration, orchestrators that guarantee correct application of Quality-of-Service policies or User policies, forms of mixing diverse distributed infrastructures orchestrators in order to gather the goods from all and to cover particular data manipulation scenarios.

- Risk assessment, which delivers methods and methodologies for building consistent security and privacy reports for assets and services to be exposed externally, methods and methodologies to build infrastructures intended to lure malicious users for organic analysis of their ways to bypass restrictions.

Research Contribution

The thesis includes both case studies and practical solutions in the area of security and privacy of complex and critical systems, including also the adoption of emerging technologies, methodologies and practices. Therefore, in the study I adopted and adapted technologies like Kubernetes, Blockchain, Containerization Engines or paradigms like shifting to cloud, grid and high performance computing or transition towards microservices architecture. The main contributions are the architectural overview and architecture methodologies proposed to overcome the security and privacy challenges in various, multi-node, multi-tenant complex infrastructures, highly scalable, covering a wide range of usage scenarios: from governmental services to finance or information technology sectors. It is counted among my contribution the following:

- proposed a methodology that gradually support the transition of complex high performance computing orchestration solutions, designed as monoliths, towards microservices paradigm
- implemented an automated workflow in one of the most expensive, critical and important High Performance Computing and Grid Computing infrastructures, hosted by the ATLAS experiment at CERN that allow tenants to properly visualize and manipulate machine learning jobs/task processed data on an on-demand basis
- provided a complex analysis on how to adapt Kubernetes to support workload in High Performance Computing, with emphasis on maintaining the user context isolated and secured
- generate a classification taxonomy for Kubernetes adaptations in High Performance Computing for recent scientific literature
- presented simple but important improvements to Kubernetes adaptations in High Performance Computing from recent scientific literature to better support the model and methods adopted
- led a study on the security and privacy of the containerization engines available on the market with emphasize to the privilege escalation problem, investigating how by stripping the isolation taxonomy may increase the resilience to corresponding attacks
- proposed a concept architecture for a system over Kubernetes that can constitute the basis for designing a HPC-Grid governance mechanism, following patterns and practices inherited form the cloud paradigm
- presented a methodology to design an e-election system over Blockchain with focus on privacy and anonymity, by enforcing a data abstraction model in three stages
- proposed a complex methodology to help architects on defining honeypots architectures over Kubernetes, alongside the legitimate production environment
- provided a heuristic and framework for defining honeypot generators to hide possible similarities between successive deployments, provided an heuristic and a architecture for defining honeypots generators in order to make such critical systems resilient to deployment pattern matching over multiple distinct infrastructures
- provided a heuristic that implements real-time transactions over slow blockchains

1. Data Accessibility

The first point to be defined regarding data privacy and security is data accessibility. Therefore it is essential to understand several methodologies of exposing data to the outside world and the mechanisms that facilitate quick reaction in case of failure, data breaches that require a hotfix, updates with high availability constraints on data access and so on. Thus, I explore the capabilities of a cloud environment to be used for displaying critical data and the opportunities to expand existing distributed processing infrastructures with emerging services. The aim is to ensure a small code change footprint to enable new services in critical, expensive but legacy infrastructures in order to cover modern data protection policies.

1.1. Research Questions

- What are the opportunities to address the necessary automation over the Software Development Lifecycle in critical infrastructures?
    - How to perform early detection for potential data exposure breaking points?
- How to permit quick reaction in case of a 0-day exploit?

1.2. Contributions

- Stan, Ioan-Mihail, Siarhei Padolski, and Christopher Jon Lee. "Exploring the self-service model to visualize the results of the ATLAS Machine Learning analysis jobs in BigPanDA with OpenShift OKD3." *EPJ Web of Conferences*. Vol. 251. EDP Sciences, 2021.

1.3. Background

1.3.1. DevOps and DevSecOps

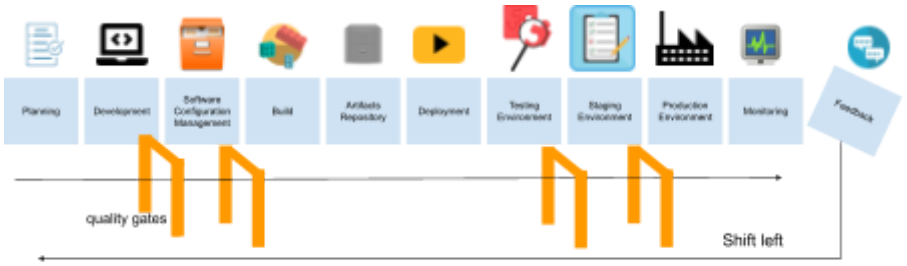


Figure 2 - Software Development Life Cycle - Data Flow

A good opportunity to sustain the digital transformation effort in order to modernize legacy software is to outsource small but relevant functions to a cloud provider. Therefore, instead of inserting the entire logic of a new feature into a huge code base, sometimes defining monolithic and

complex structures, a way to improve the development process is to detach them from the main solution and design them as individual services or even microservices. Therefore the effort to respond to change requests comes down to only defining loosely-coupled remote routines and not embedding the entire new logic into the complex structure of the system. In order to sustain a significant growth in modernizing the legacy infrastructure with such an outsourcing management model, it requires to enforce an automation mindset among the contributors to digital transition. One step further is to adopt methodologies and practices from the DevOps culture and to generate automated pipelines (assembly lines) following the Software Development Life Cycle data flow (Figure 2). One important practice, essential to modern software development is Continuous Integration[45]. By definition, continuous integration covers the software delivery process from the code base up to the deployment in the integration testing infrastructure (Figure 3).

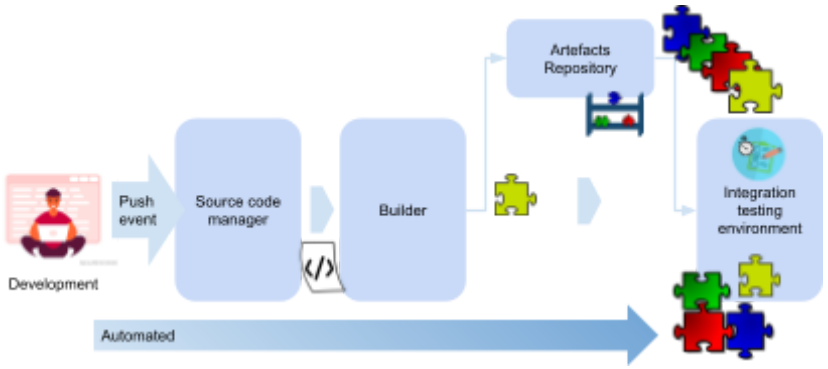


Figure 3 - Continuous Integration

One quality gate can be set at code submission time and is achieved by performing static code analysis before going further and triggering the build stage. At this point in time, a pipeline enforcing code analysis may identify anti-pattern code snippets and may propose coding standards, with the aim to improve not only the efficiency, but especially the security and the privacy of data exposed. For example, as P. Ferrara et. al. present in [46], in such an early stage, one can have a consistent look on the privacy requirements over the GDPR policy. While passing the quality gate, the pipeline can trigger the build stage that, in many cases, must allocate physical resources in order to compile executable artifacts. With the evolution of containerization systems and with the mass adoption of the cloud paradigm, other buildable artifacts are also the container images. Containers, for their ability to run on heterogeneous environments and for their intrinsic portability, are an alternative for delivering software, very similar to any package management system.

Once constructed, all artifacts (binaries or container images) must be stored in dedicated artifacts repositories. From here, sequential deployment stages will download the executable objects and put them in running contexts. In addition, the build stage can integrate another security gate represented by the execution of the unit tests. If passed all unit tests, the piece of software is qualified to go further to the next stage in the pipeline.

Last step in Continuous Integration is the Integration testing. Here, if part of a larger solution, each piece of software previously built is tested together with all the other applications or services in order to identify any participation issue inside the ensemble. Thus, while in the previous stages, the artifacts were tested individually, now they are validated together as a whole. At the end of this session, a deliverable is generated that needs to go through an acceptance testing stage, before being ready to be launched in a production context. In all intermediate levels, an important part of validating the maturity of each artifact is to verify them from multiple angles. Therefore, an important insertion between these regular stages is a rigorous security and privacy testing. Here the emphasis must be on data leaks, privilege escalation, vulnerabilities in the libraries used and so on. Therefore, as a maturity

metric, the software must be tamper-proof to at least the well-known vulnerabilities and attack methods, and not just released in the wild.

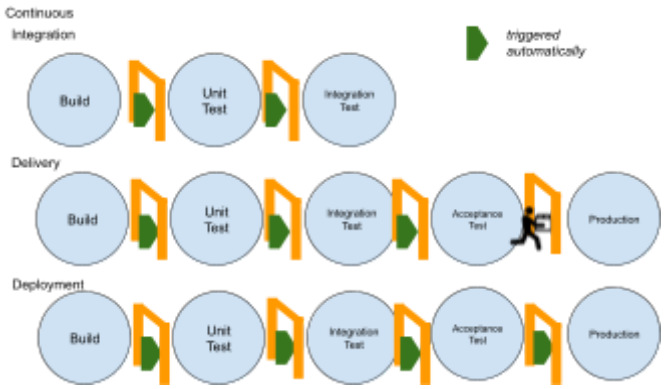


Figure 4 - Continuous Integration, Continuous Delivery, Continuous Deployment

Once the acceptance stage is fulfilled, it has to be decided if the solutions developed can be automatically deployed to production or require human intervention (Figure 4). This thin border between the two approaches delimits two common practices, namely continuous delivery and continuous deployment. One relies on continuously producing viable products, while the other assumes the risk of a continuous flow of development and launches into production every change that has passed the quality gates set. This latter step is also evaluated automatically. The term continuous implies that in all targeted stages, the transition must be made automatically.

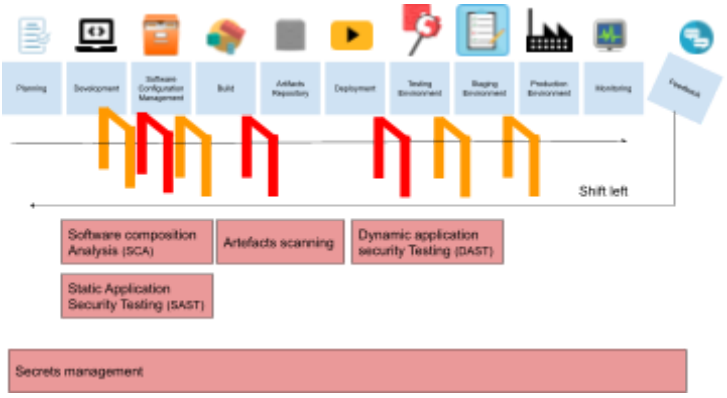


Figure 5 - From DevOps to DevSecOps

As DevOps [52], DevSecOps is more of a culture trend in the IT industry supported by tools and automation and the mindset of continuous learning and continuous improving the streamlines used internally as part of the Software Development Lifecycle. Pragmatically, DevSecOps impose a shift left of the security (and privacy) validations with the aim to overcome the bottleneck bred by the security teams while assessing deliverables prior to moving them to production. As can be seen in Figure 5 and defined by R. Kumar [53] et al., particular security and privacy analysis can be performed on each lifecycle stage. Thus, the security teams must make a selection of automatic tools, set the desired security and data privacy constraints and policies in the context of the deliverables produced and train the developers and operational teams in relation to the working principles. Such multi-dimensional approach speeds up the time to market and provides the premises for producing software development in continuous flow, with deliverables that are presented in a mature and consumable form. Software development is fueled by various security and privacy analyses stages:

Source Composition Analysis (SCA), Static Application Security Testing (SAST), Dynamic Application Security Testing, Secret Management and so on, while Operations teams focuses on hardening, runtime security, monitoring and so on [53].

1.3.2. ATLAS Infrastructure at CERN

Each of the principles and practices presented in the current section were considered in the development of a feasibility study (proof-of-concept) regarding the digital transformation and expansion of the ATLAS distribution infrastructure at CERN. The effort made was to onboard new functions via the cloud paradigm with a minimum of effort required in terms of changing the complex code base built over many years of existence. The focus was to propose new architectural models such as the principle of distributed services and microservices and to perform automation throughout the development cycle of new features. The proposed solution has been created to support the transition towards the DevSecOps principles and mindset, especially from the perspective of the infrastructure's criticality and its importance in a global context. The proof-of-concept was built around a solution for visualizing data processed in the Grid and HPC infrastructure for machine learning tasks, since, in an internal context, they require a special management and a dedicated handling methodology with the aim of privacy and multi-tenancy. The cloud enabler was the OKD version 3 (a Kuberentes flavor for enterprises, powered by RedHat OpenShift)

Before going further, I will briefly introduce the structure of the ATLAS computational infrastructure dedicated to simulations and analysis tasks.

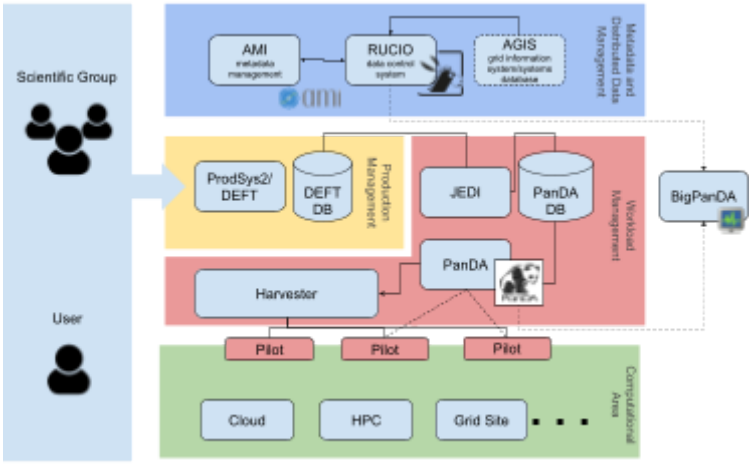


Figure 6 - ATLAS Distributed Computing (Analysis infrastructure) High Level Overview

The ATLAS distributed computing system grants scientific groups and individuals with the capability to analyze huge and expensive collections of data generated by particle collisions in the Large Hadron Collider (HLC) and detected as events in the ATLAS sensor at CERN [55]. At the same time, for the validation of scientific reasons, the same infrastructure serves for running Monte Carlo simulations [56] and also for executing distinct production scenarios, originating from other scientific areas. As can be seen in Figure 6, compiled from the information and architectural views present in [57][58][59], the infrastructure abstracts and orchestrates scientific requirements, hiding behind a complex data, metadata and workload management system. The ensemble is capable of running over heterogeneous systems, distributed over large geographical areas and implementing different paradigms (HPC and Super Computers, Cloud, Grid, University Networks and so on.). The system itself embraces the Grid Computing paradigm, tailored to support the variety of systems involved in scientific research. The orchestration of analysis or simulation tasks/jobs receives various inputs for

their optimal scheduling and execution, including here the problem of data localization, which can be in huge quantities, difficult to move to the premises of another Compute Engine. Other obvious problems are those regarding the execution constraints and user policies applying on scientific groups, individuals, or the resources availability.

The brain of the tailored infrastructure is the PanDA workflow manager [58], which gathers around it all the essential systems. It implements the batching system, taking over the standardized tasks from the submission platforms and queuing them in priority queues. Based on the input from the metadata providers, it communicates with the Harvester buffer system or directly with the Pilot systems to forward the tasks/jobs to be executed by the processing engines. PanDA is equipped with a system for monitoring tasks/jobs throughout their lifetime, making it a perfect candidate for the pilot project proposed by me. The monitoring platform is called BigPanDA [59].

1.4. Case Study - Data visualization cloud service with multi-tenant support for HPC and Grid Processing

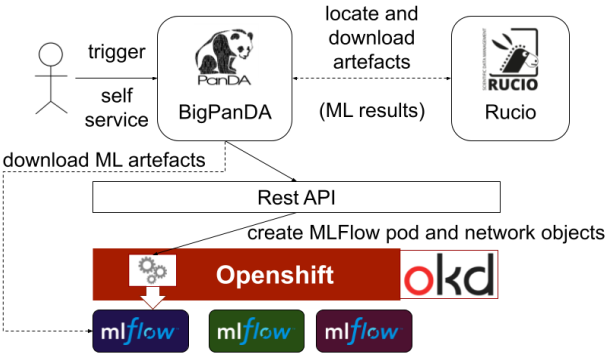


Figure 7 - Concept architecture of a self-service system for ML data visualization services

In the current case study [60], I focus on defining the interaction between BigPanDA and OpenShift OKD. The context of such an interaction is related to the need to view Machine Learning analysis data in a friendly format. At the same time, a management model detached from the core ATLAS orchestration is being tested. As can be observed in the concept architecture and interaction flow (Figure 7), a tenant, which has previously submitted a Machine Learning job or task to the ATLAS processing infrastructure, can also request a visualization service to display the results. For each demand, the BigPanda controller triggers the creation of a web service in OpenShift OKD. As part of this routine, BigPanda's role is to locate the results and download[59] them from the ATLAS distributed infrastructure, via Rucio[72]. Once data is stored and indexed within BigPanDA, the controller calls the OpenShift OKD API and triggers the creation of an MLFlow web service along with all the OpenShift (Kubernetes) communication and configuration objects. The MLFlow pod downloads the Machine Learning artifacts from BigPanDA and stores them locally, in a temporary/volatile location.

Following such an approach, the BigPanDA solution outsources data visualization management to an external cloud platform, optimized for this type of interaction. The principle on which I based the architectural model is "segregation of duty". Therefore, instead of having one solution that fits all, various auxiliary routines can be detached from the main solution and executed through specialized platforms. OpenShift OKD is a viable platform for various scenarios, especially when it involves creating on-demand services, multi-tenancy and intelligent container management.

To take advantage of OpenShift and also to increase the portability of my solution, I deliver the MLFlow instances in containers and pods. Following this cloud native model, the solution is extremely portable and can be easily adjusted to run on most public and private clouds.

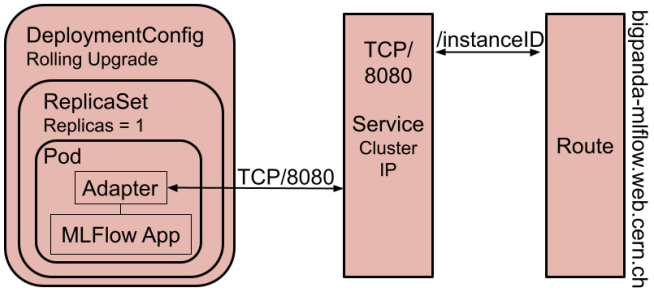


Figure 8 - OpenShift objects and communication model

OpenShift OKD is a Kubernetes implementation, therefore, it inherits the same base architecture as the one implemented by the core project. A service that needs to be exposed to the outside world will most often be accessible through the ingress-managed load balancer. Therefore, if a client would want to connect to a service running inside an OpenShift OKD cluster, the client will be able to use a specific domain name to reach that service. Furthermore, an ingress controller can also manipulate internal routes to forward traffic toward various services based on subdomains or paths selectors. BigPanDA will use the routing mechanism (ingress controller) to enable multi-tenancy by creating unique fan-out definitions (Figure 8) and communication primitives for each MLFlow instance, through the OKD API.

In addition, I observed that both OKD 3.11 and MLFlow v1.9 (versions selected for the proof of concept) do not support target rewriting. This concept means that if an HTTP request path does not correspond to an existing resource or application endpoint, the request will end up generating a resource-not-found error. Since I will be using one DNS domain for all MLFlow instances running in parallel, one can identify each instance by a random string embedded within the resource path. Some web servers or web applications support remapping a non-existing endpoint to the root path. However, both out-of-the-box OKD 3.11 with HAProxy Ingress Controller and MLFlow v1.9 do not have support for such configuration. To work around this issue, I adopted a multi-container design pattern for pods - the adapter pattern[73]. An Adapter object is located in front of the main MLFlow application service, and it handles the requests coming from the load balancer. Each HTTP request will be translated and sent to the MLFlow service through the localhost interface (Figure 8).

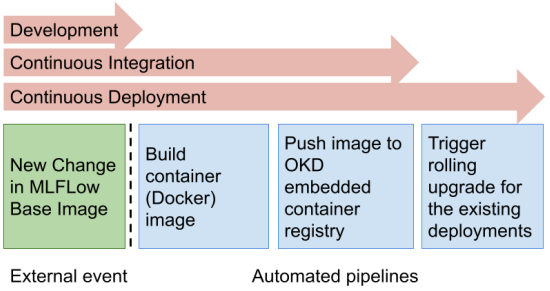


Figure 9 - DevOps practices

Finally, and also part of the architectural vision, I decided to create my own base container. It includes the MLFlow middleware and the assembly scripts and is built using the native build

capabilities available in OpenShift OKD. Therefore, I separated the preparation of the base MLFlow image from the actual application deployment, respecting the DevOps methodology and practices (Figure 9). All the configuration items are currently stored separately from the BigPanDA code base, and everytime a change is detected, a trigger executes the Continuous Deployment pipeline.

1.5. Insights

In this current case study I have developed a new feature for BigPanDA that offers the ability to visualize the machine learning results, produced in the ATLAS Distributed Computing infrastructure. A tenant (PhD student, scientific groups etc.) can request such a service directly from BigPanDA, and the workload will be further delegated to an OpenShift OKD cluster via REST API Calls. OpenShift will spin-up a MLFlow pod per request, download the Machine Learning artifacts from BigPanDA, expose the web service to the outside world and maintain high availability through the native healing mechanisms. Following this delegation model, I demonstrated that BigPanDA can easily adopt a cloud native approach and also that it can function as a catalog of scientific services, in the context of the ATLAS Distributed Computing at CERN. Moreover, I also followed several DevOps methodologies and practices to facilitate the build of the base MLFlow container from scratch. Therefore, I implemented continuous integration and continuous deployment pipelines using the native mechanisms of OpenShift OKD. These pipelines are triggered automatically each time a PUSH event occurs in the external configuration items repository. Furthermore, as a good practice, the deployment strategy is using the rolling-upgrade model, a method that minimizes downtime inevitable for an upgrading process. Finally, I also implemented a garbage collector, a Python procedure that identifies old service instances and deletes them if they passed the 24 hours expiration time.

The results obtained during the testing phase certifies the end-to-end functionality of the integrated solution. For new iterations, I have already identified a few other paths of development and optimization and here I include: the use of initialization containers, a migration to the operators pattern, use of health probes and the delegation of the routing process to another ingress controller solution. These changes can bring significant improvements to the solution architecture, as they apply a more performant functional block separation model and also remove the need to have additional components such as an adapter object.

In the current case study, I presented a way to expose critical data in complex, legacy, and multi-tenant environments by outsourcing the data visualization function to an external cloud environment. An easy way to restrict access to data, accessible to any developer, is to implement light obfuscation methods over the access coordinates. In addition, automation plays a key role as a method to react quickly in the event of a major hazard. All these implementations are relevant components in establishing the data accessibility dimension.

2. Data Anonymity

In any data exchange, the transmission infrastructure needs to ensure no critical or private date is disclosed. This is a mandatory security and privacy function and is frequently added as a native capability to communication facilitators. From asynchronous HTTPS API calls among services running together to provide complex outcomes to securely encrypting and signing data prior to storing it to a distributed, decentralized database, many technologies cover such needs and provide them with no extra effort, while deploying the solutions. However, such capabilities only establish boundaries that enclose the information transmitted among end-entities and do not obfuscate the source of data or the information that targets entities or individuals. Thus it ensures only half of the privacy needs when performing analysis on critical data. The other half has to deal with the actual knowledge shared and

must hide any private information of the origins of data or of the targeted audience [75][76]. Blockchain provides peers with mechanisms to hide their real identity behind a digital identity. However, as Q. Feng et al. present in their state of the art paper [80] , through its peer-to-peer nature, a public implementation developed around crypto-currencies is not fully protected against network analysis or address clustering analysis that may reveal the source of data. My approach is to adapt and distribute the effort of handling data to multiple institutions in a private and permissioned implementation, while part of the anonymity and obfuscation heuristics and functions are outsourced to other, outside of blockchain (off-chain), custom technologies. I demonstrate that blockchain solutions can cover the needs of a critical governmental process - the state election process, following and aggregating success stories from countries that ran such e-election pilot projects before.

2.1. Research Questions

- How can ensure anonymity/information confidentiality over public[1] data in critical systems that require user authentication?

2.2. Contributions

- Stan, Ioan-Mihail, Ilie-Constantin Barac, and Daniel Rosner. "Architecting a scalable e-election system using Blockchain technologies." *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2021.

2.3. Case study - Blockchain in government services

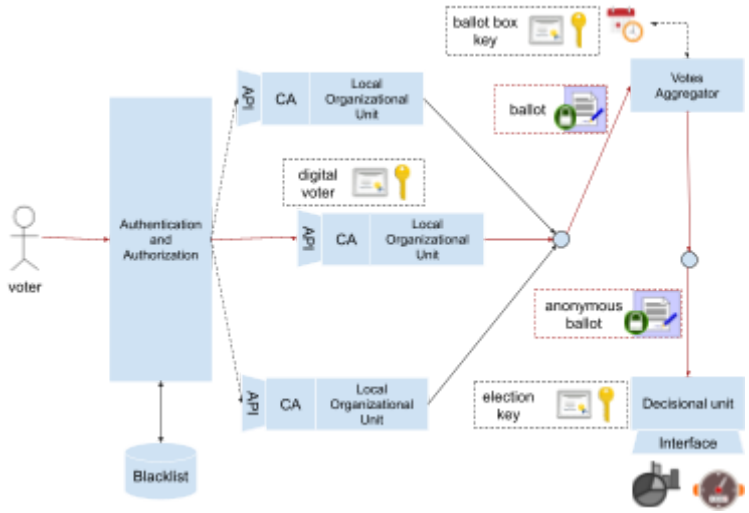


Figure 10 - E-voting platform - concept architecture [92]

When defining the architecture of an election system [92], it is mandatory to embed security patterns within its structure, from the very beginning. The main challenge when running a nationwide e-voting system for state or community elections is to ensure a balance between anonymity and the risk of inserting malicious unsigned ballots within the system. Since these 2 concepts may somehow conflict with each other, the activity of splitting the architectural view into dataflows and fully

[1] shared/exposed outside users' premises

isolated functional blocks becomes a de facto requirement. Therefore, when designing an e-voting system, one main aspect is the segregation of duty, on one side from the business logic perspective and on the other side from the target responsibility group perspective. One state organization should evaluate the authentication and authorization part and translate an active elector into a digital entity, another state organization should deal with the flow of the anonymized ballots, double checking and removing any personal information and transferring them to a collector entity. Finally, the election process should be concluded by a specialized organization that needs to reveal and count the votes and publish the results.

To ensure anonymity, the authentication and authorization mechanism needs to be fully detached from the main voting system (Figure 10), thus minimizing the specter of chances to be able to match a human being with its digital identity. The authentication and authorization mechanism uses the legal means to identify a person that has the right to vote in the current election process. Furthermore, based on the user coordinates, it also has the role to allocate the digital identity of a voter to a local organizational unit for optimizing the election process and increasing the traceability in case of malicious attempts to generate multiple ballots (a double spending problem [96]). This heuristic needs also to monitor those users that were blacklisted by the legal systems, and it has also to be in touch with the local organizational units to revoke those certificates that are connected to a recent blacklisted user, and in consequence to cancel all the votes generated by that user.

Once translated into a digital entity, a citizen gets a private key and a certificate signed by the local organization, which can be used in the encryption process to sign digital ballots for participating in the current election process.

The voting process itself requires 2 encryption stages and should allow participants to change their option before the election due date. The ballot itself will be wrapped into a “double envelope” [97] to secure the information and protect the identity of the elector during the election process. In the first encryption stage, the information stored within the ballot will be encrypted with the public key allocated for the current election campaign. Once secured, the ballot must be delivered to a collector unit, dataflow which triggers a second encryption process. In this second stage the output previously generated is signed with the user’s private key and sent to a collector unit, the information being encrypted with the collector’s public key. Pragmatically speaking, the ballot box has its own pair of keys with the aim of hiding and protecting the ballots in the digital context. However, in case the ballot box is compromised, the participant’s options won’t be visible since the election campaign private key will be hosted by another unit which is in charge to compute the voting decision.

The aggregator or the digital ballot box will continuously receive transactions containing the voter’s options, preserved in a cryptographic envelope. The entity will actively monitor the time frame established for the voting process and consider only the latest transaction received from a specific origin. Therefore, in case an elector will want to change its choice before the election due date, only the latest option will be considered. Once the time frame allocated elapses, a routine will reiterate through all ballots received by the collector entity, will extract the encrypted recording of a vote, and remove the user’s signature. At this point in time, any link to the originator of a vote will be removed. The collector entity will forward all the encrypted ballots to a decisional unit that will further compute and publish the results of the election. Any new transactions coming from local units will not be considered and will not be further sent to the decisional unit, making it impossible to manipulate the system after the election’s time frame elapses and the digital ballot boxes are sealed. One key aspect when defining such a model with 2 main dataflows (local hubs to collector unit and collector unit to decisional unit) is to isolate the communication channels between local organizational units and decisional units. Following this approach, the information flow will be hidden for those critical components of the system and therefore no unit will be able to run complex algorithms and do parallel calculations based on behavioral patterns. Of course, if all the system’s components will be managed

by only one authority and the digital security will not be doubled by law, the system will not be tamper-proof. However, by distributing the responsibility to specialized administration units, such segregated topologies may become opportunities in designing nationwide e-voting platforms.

The structure proposed follows some success stories and design patterns absorbed from existing implementations from countries like Estonia, Norway, or Switzerland. However, one interesting challenge remains the structure of the authentication mechanism. Here one implementation has to bring to a remote location or into the elector's home, the security and privacy provided by a specialized voting center. An option can be the adoption of a specific election card, or a key stored in an electronic ID, hidden behind a pin code, and read with a specialized USB chip reader. Another option can be a digital signature obtained from a valid organization prior to the voting process.

A third option, and the one that has been considered by me when designing the concept architecture, was the idea of obtaining the full control of the webcam of the device from where an elector decides to express the voting options and use this capability to gather essential information. Prior to accessing the voting interface, a user is asked to scan the ID or passport, by using the device's webcam and to pose for some profile pictures [98]. A similar method is currently used by companies that provide alternative banking services [99] to open new debit accounts on behalf of a person. Once obtained all the required input data, complex artificial intelligence algorithms can ingest that information and authenticate or reject the access of an user by trying to find similarities between the identity retrieved from the official documents and the pictures taken by the webcam.

Another key aspect when detaching the authorization and authentication mechanism from the main e-voting platform is the ease of management when the election process runs a hybrid model. Having one centralized system connected to any evidence of population databases and able to register people that express their vote through any means may also significantly reduce the risk of double voting. This model needs also to be supported by a methodology to establish the priorities in case of an attempt to vote through all available means (e.g.: votes expressed through postal services vs electronic vote that can be easily canceled through certificate revocation; paper ballot must replace any other method).

As the global context has shown us the stake of digitalization during the Covid-19 crisis, quick solutions for complex problems might be an important driver nowadays. One big opportunity is to reuse existing assets in solutions and scenarios where they can bring a significant contribution even if they were not designed to perfectly fit within the context.

Considering this driver, I adopted Blockchain and Hyperledger Fabric to make use of those well-tested and mature routines that address part of the critical building blocks proposed in the architectural view.

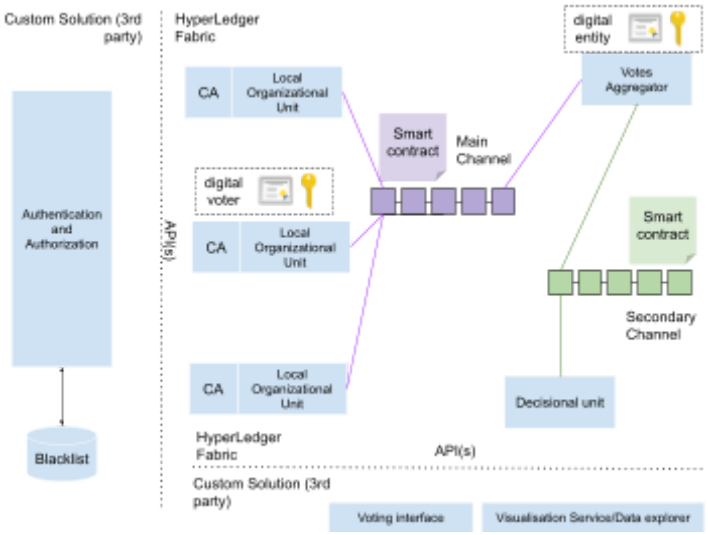


Figure 11 - Designing over Hyperledger Fabric [92]

The model adopted by Hyperledger Fabric [36] emphasizes the role of organization in the structure of a communication topology. Each macro entity is represented by an organization on behalf of each, users can send information within the network. One organizational entity can establish its own validation rules and admission controllers and can also simulate the behavior one transaction may trigger within the network. Furthermore, the structure can be distributed to several systems, not necessarily collocated, which can also segregate the responsibility for specific decisions to specialized administration entities. Having a broader look (Figure 11), besides the participative component, one organization has its own Certificate Authority to authorize and validate users subscribed to it. For my use case, I have 3 types of organizations, but only one type exposed to electors to express their vote. As previously described, I have several local hubs that can be distributed geographically, one collector unit or a digital ballot box and a decisional unit that needs to receive all ballots and to be able to compute the election results. Since the aim of my architectural model is to centralize the authentication and authorization mechanism, the user's distribution, and communication with local organization's CAs will be managed from external via Hyperledger APIs. Thus, one actor will go first through a 3rd party authentication mechanism that will take decisions based on the actors' coordinates. First, a heuristic will assign the user to a local organization (e.g., based on the user's home address). Once decided, the same external entity will trigger the appropriate Certificate Authority to generate a digital identity for the authenticated user. The same entity will monitor the blacklisted users and ask the organization to cancel and to remove the rights of voting for a specific identity.

Once authenticated, the users will get access to another 3rd party interface, associated with the local hub. From here it will be able to generate encrypted ballots and trigger blockchain transactions on behalf of the local organization. Thus, the additional logic that stays above the existing routines of Hyperledger Fabric, will be the ability to generate a secret ballot encrypted with a supplementary public key, generated in advance for the current election process. Once generated, this value-bearer asset (in the form of an encrypted message), will be sent through blockchain to the collector entity (digital ballot box) in the form of a blockchain transaction. At this point the ballot will have a double envelope, since it is in the form of an encrypted string, encrypted also by the transaction mechanism with the key belonging to the collector.

The transaction flow process can be redesigned via custom chain codes, since it implies several constraints related to the enforcement of a time frame for the election process and the ability to change the original vote before the due date. Thus, the chain code will require to interrogate an

oracle to monitor the time frame (e.g., a Network Time Protocol server from outside) and must override the value-barer asset in the destination if several other transactions are triggered from the same source. Moreover, if the time frame elapses, the chain code should stop processing any further transactions.

Another chain code might be also required between the collector entity and decision-making unit. Once the election time frame ends, this chain code should gather all the assets received by the collector unit, remove any digital signature, and send them in transactions to the decisional unit account. Once reached, the decisional unit can forward the information to another 3rd party service or run another chain code that hosts the current election key pair and decrypt the ballots and compute the results.

To provide full isolation between the communication channels established among the local organizations and the collector and between the collector and decisional unit, Hyperledger provides the concept of channels. Therefore, an implementation using this feature will have two different blockchains and two chain codes residing on different blockchains.

Since Hyperledger Fabric is a permissioned/private blockchain technology, the consensus mechanism can be simplified based on the business need. In my case, the consensus should be established by the receiving unit, like a supply chain implementation. Therefore, the destination organization should validate the parameters of a transaction and propose blocks.

Using blockchain may also increase the traceability in case of an investigation of fraud as it will be easier to follow each stage a vote has been through, even if the information recorded is secret.

2.4. Insights

In the current case study [92], I proposed an architectural model, oriented on blockchain technology for a national voting system. The design and architectural decisions were built bottom-up, starting from the Hyperledger Fabric structure, and incorporating patterns from existing voting systems. The novelty proposed by me results from the model of segregation of the functional components, from the authentication method taken from the world of alternative financial solutions and from the simplicity of the model easily adaptable to any state regional organizational chart. In addition, a minimal implementation of the e-voting platform provided a good prediction of scalability. I based my measurements on fine-tuning of the Hyperledger Fabric configuration parameters, and I identified those configuration items that will increase the overall system performance - blocks with many transactions and embedded GoLevelDB database for state management. Following a distributed and decentralized model, the system proposed turned out to be fault tolerant and resilient to malicious forms of manipulation.

The transaction heuristic chosen, by establishing 3 phases of the voting process, the aggregation and generalization stange enforced and the use of the double-envelope methodology, offers a granular perspective on the establishment of the dimension of the anonymization of data transmitted.

3. Data Transportation

Another essential aspect to be presented in regards to data privacy and security is how data is moved while properly maintaining the boundaries of ownership and its secrecy. Such aspects may require better understanding of how a communication topology must be built in order to cover natively such requirements. Furthermore, it is also very important to understand the opportunities of scalability and the resilience of the infrastructure deployed in the event of a network attack. While many infrastructures are able to properly manage privacy and security demands, sometimes they may

involve considerable performance or resource demand costs which must be balanced in order to provide a desired outcome. In communication in particular, there must always be a tradeoff between security and privacy policies and velocity, since the excess of one affects the other dimensions. Full security and privacy analysis on single or correlated events, captured in a communication network, may bring a disproportionate cost of resources and may slow down the communication flow.

3.1. Research Questions

- How to ensure data authenticity[2] and change data ownership in byzantine[3] communication?
- What are the opportunities, and which are the costs to transport confidential and valuable data?
    - How to transfer valuable digital assets over the Internet?

3.2. Contributions

- Popa, Alin Bogdan, Ioan Mihail Stan, and Răzvan Rughiniș. "Instant payment and latent transactions on the Ethereum Blockchain." *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2018.
- Stan, Ioan-Mihail, Ilie-Constantin Barac, and Daniel Rosner. "Architecting a scalable e-election system using Blockchain technologies." *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2021.

3.3. Background

Besides its storing function, Blockchain is a peer-to-peer network (P2P) that ensures communication between peers via transactions. Thus, each transaction (communication occurrence) is validated by the community and stored, in data structures called blocks, in the ledger, for traceability. The validation process requires recipients to satisfy the conditions imposed by the smart contract in order to process the transaction. Once validated, a transaction is performed only when the consensus was reached among participating peers and the transaction is listed in a newly generated block further attached to the blockchain. While performing such a complex validation process, the times of processing transactions may increase and therefore the communication can be less efficient. However, depending on the implementation, different blockchain solutions can record different time values between consecutive block promotions. In the public spectrum, implementations for cryptocurrencies are directly impacted by the consensus mechanism. In Bitcoin, for example, the time between promoting consecutive blocks is approximately 10 minutes [109], while in Ethereum, the GHOST protocol[110] supports quicker block times, approximately 10 to 19 seconds. In addition to transferring digital assets between peers, a blockchain solution has to deal with exchanging transactions and blocks information among all peers. Furthermore as V. Deshpande et. al [106] explained in the problem statement, each peer needs to maintain locally information about the communication topology, full or partial. Therefore, each individual needs to rely on peer discovery

[2] origin
[3] exposed to a wide spectrum of attacks

mechanisms, filtering mechanisms for inbound traffic, neighbor selection heuristics and so on. Part of these side actions have a direct influence in the overall communication progress. As V. Deshpande et. al emphasize, the communication topology properties is influenced by the blockchain type, the consensus mechanism, the amount of inbound and outbound connections and by the neighbor selection process. In addition, the paper provides measurements that prove how permissioned blockchains can manage fewer peers than public blockchain implementations, due to the lightweight consensus model. Furthermore the authors make a distinction between voting consensus mechanism and lottery mecanism, the latter being able to gather more stakeholder inside the network.

As J. Spasovski et. al. defines, compared to non-blockchain implementations [108], blockchain solutions may have delays up to 2-4 times in response time and may not have the same resilience in case of heavy load. In order to minimize the impact of the consensus mechanism, authors run analysis on proof-of-stake blockchains, lightweight consensus mechanisms that conserve the processing power and engage peers to participate with their wealth in the network. Since their stake is directly impacted by the behavior shown in the distributed and decentralized system, it requires less power and causes less congestion. However, as the authors emphasize, blockchain solutions scale linearly and with enough nodes joining to the network, such implementation may still record good telemetry: high throughput and low response times. With the intrinsic values of blockchain regarding immutability it may still be a perfect fit in various scenarios including critical industries focused on secured transactions.

Blockchain became a suitable option for ensuring a proper and secure internet-of-things communication platform, supporting big data, especially in the private spectrum. As J. Zhang et al present in the paper [107], blockchain implements the concurrent communication tree model and also is able to distribute the storage function towards multiple aggregator nodes, enforcing also proper security and privacy over data transmitted. In private and permissioned blockchains, users can adhere to organizations' networks in order to gather information from sensors and IoT devices [111][112]. Thus it can be used in smart cities implementation to gather relevant telemetry and to securely share it to the audience. Furthermore, it can be used to trade energy in microgrids, sustaining the innovation of the energy industry.

Public blockchain are the most well known implementation by masses and are mostly used in cryptocurrency. With the rise of Ethereum and their turing-complete smart contract programming language [113] - Solidity, a new age of online services consumption has been born. Thus, Ethereum, through DApps [114], is now able to provide processing power on demand with no need to own the underlying infrastructure by one service provider. However, over time, blockchain solutions were not fully tamper proof against network, smart contracts or consensus attacks [115][116][117] and sometimes such attacks resulted in actually corrupting the ledger. One of the problems is related to the double spending problem, when, in cryptocurrency, a peer fools the network and manages to spend the same value-bearer token in two distinct transactions. Two of these transaction security issues [115] are represented by the Finney Attack and Race Attack [117], both making use of an early logic flaw in processing transactions. In recent updates, such issues are no longer possible, since now transaction processing is constrained by a confirmation heuristic. The most problematic issue, also related to transaction security, is the 51% issue [117]. This happens if a peer or a sub-community of peers control more than a half of the network. These cyber-cartels, in theory, can alter the blockchain as they are capitalizing more hash rates than the other miners.

The spectrum of network attacks is also comprehensive being the most relevant in the context of communication. Categories of attacks are also inherited from former P2P solutions, used in other domains. One relevant problem is the Sybil attack where a malicious user can create and hide behind multiple identities. Another relevant network security issue is the Eclipse attack [115][116], a targeted attack with the goal to disconnect a node from the blockchain and make it communicate and exchange

information only with malicious nodes. Such issues can be solved if the node is able to identify legitimate neighbors and reject any other inbound traffic.

In addition, as in any publicly exposed setup, all participating nodes can be subject to Distributed Denial of Service attacks, therefore with no extensive protection in place, targeted nodes can be disrupted.

Part of the security issues previously presented may not be relevant in private and permissioned blockchains, as in most cases, these are closed networks where the peers are certified priorly, before joining the community. Public blockchain are permissionless, meaning that everyone can join and any information exchange must be validated by miners which involve enough processing power to protect the network. All the information is public and everyone can read from and write to the distributed and decentralized ledger. Private permissioned blockchains may fit better to other use cases, where the concern is not on complex validation but on traceability inside closed communities. The cost of running consensus mechanisms like Proof-of-Work, able to cover the full byzantine context, may not be feasible, therefore lighter consensus mechanisms are required. Algorithms similar to RAFT[118] implementing the leader election paradigm or similar to lottery-based election algorithms provide sufficient trust in private-permissioned blockchains.

3.4. Case study - Instant payment in crypto-financial transactions via the Ethereum Blockchain

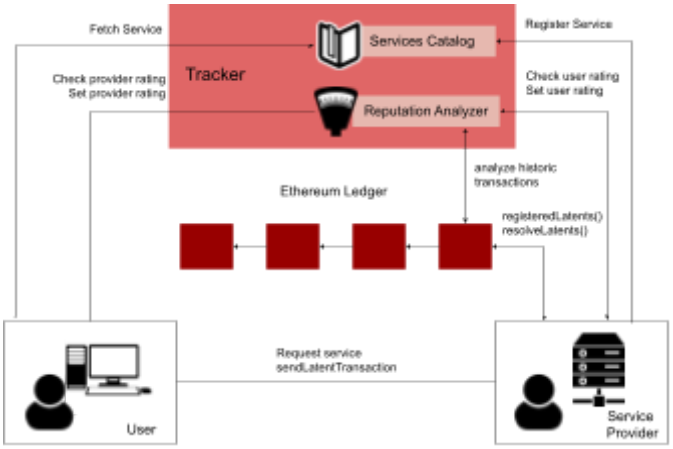


Figure 12 - Latent Transaction System Concept Architecture

Implementation like Ethereum [120], which provides mechanisms to extend the base functionality of the network via Turing-Complete smart contracts and DApps, opened up a new market for managed services, renting processing power in exchange to crypto currency. Therefore, through DApps, in theory, one can access a large variety of services developed as an extra layer over the public setup. Even if Ethereum records small block times, for real time services still does not offer good perspectives. Therefore, one method to overcome the velocity issues is to develop a complementary infrastructure in a hybrid mix. Therefore the innovative idea on which I contribute is called latent transactions [119] and implement core banking and insurance services over Ethereum. A latent transaction is an off-chain agreement that provides guarantees to both consumer and producer/facilitator that in exchange for real-time services, a remuneration transaction will be processed at a later time. The two guarantees on the consistency of the agreement are the ERC20 [121] standard for developing smart contracts that provides the means to control the balance of a peer once a contract is established and an alternative rating system for both stakeholders, based on previous experience (Figure 12). Furthermore, in addition to novel latent transactions, the hybrid setup also

proposes a platform that displays a service catalog, ready to onboard new real-time service providers (as for example game servers maintainers or video streaming service providers).

Following the industry standards in regards to real time services there were established five distinct acceptance criteria for remuneration systems:

Q1 - The solution needs to be able to expose an infinite variety of services or operations

Q2 - The trust in a supplier must be based on rating and analysis

Q3 - The services payment must be adaptive based on to demand or computational effort

Q4 - The system must support instant payment/remuneration

Q5 - The system needs to accept requests and provide services even if the stakeholders are disconnected from to the public blockchain while exchanging data

To summarize, the case study presents a solution that aims to solve the transaction latency in Ethereum Blockchain and similar networks. Therefore a novel concept has been presented called latent transactions that outsource the amount exchanges to an off-chain platform while the actual payment can be registered in the main Ethereum chain at a later time. The setup supports adaptive prices and is designed for real-time services vendors. The solution has been developed following the ERC 20 smart contract standard and in order to overcome possible double spending issues due to the off-chain character of the solution, it also implements a rating service to express the trust of peers.

3.5. Discussions

In regards to the case study presented in the previous section - Data Anonymity, Permissioned blockchain represents a subcategory of the blockchain implementations that are designed to run privately and to provide a network with an extra layer on top, for access management. Therefore, each peer joining the network has to obtain a digital identity from the blockchain manager, provided via x509 certificates. Each certificate embeds a set of parameters that define the permissions a stakeholder has within the network. Such security models provide better privacy than the public solutions, since each member has granular access to the information stored into the blockchain and also has a dedicated and pre-established role within the network. However, since the participants are all known and pre-validated, such solutions lack anonymity. Furthermore, the consensus and data validation is handled by a pre-established set of participants [122]. Thus, in very specific cases, this comes with an issue as designated stakeholders can change the consensus mechanism, membership policy or smart contracts lifecycle[123] at runtime. In most implementations, in financial sectors and not only, such blockchain networks are maintained by consortium of organizations, exchanging transactions and valuable information, validated by authorized members and not by miners as in the public spectrum. As M. Cash et al [124] mention, not all peers have the right to read or write to the blockchain. By its characteristic of being private, it does not mean that a blockchain cannot be exposed outside, it means that not all peers have the access to manipulate data within the ledger. Other opportunities associated with permissioned blockchains is the semi-decentralization, where by design, it does not need broader communities in order to secure the network, but fewer participating entities. This makes it run faster and imposes lighter consensus mechanisms. Furthermore, by its nature of being modeled at runtime according to the goals of the participating entities or organizations, the solutions implemented in the private spectrum are highly customizable and provide interoperability mechanisms. However, even if in terms of communication, it implies a trust model and access management suitable for covering most privacy concerns, such solutions have their own pitfalls. Thus, by limiting decentralization, brings opportunities to corrupt the network since the security is handled by fewer, designated, peers. If compromised, they can easily affect the integrity of data. However,

such situations are less probable as most of the implementations are not transiting towards centralization but follow an incremental decentralization.

The blockchain solutions are suitable for providing the communication topology for trading valuable information, exchanged in critical or noncritical industries, especially when using permissioned blockchains. However, one pitfall is the fact that it may involve consistent computational power and storage capacity for only one service that ensures communication and traceability. Thus in many cases, the limited spectrum of services delivered in contrast to the infrastructure involved may not be feasible, as the cost of maintaining such a network may not be advantageous. Even if through complex smart contracts one can extend the spectrum of consumable services, they are still limited by slow processing, by constrained development and code maintenance overtime and also by the intrinsec immutability, which does not allow hotfixing existing contracts[125]. Therefore, for the same resources involved one can implement the cloud paradigm which can significantly extend the services portfolio and also provide various communication services that can be used (alongside the others services). Another problem regarding Blockchain is related to the GDPR compliance [126], where the right to be forgotten can not be easily implemented as long as historic data must be kept inside as part of the immutability attribute. Normally, the proof of burn which is the capability to delete the cryptographic keys which provide the access to the private data is not fully compliant with the regulations since encrypted data are still available and shared among the participating nodes. However, in hybrid setups, the real data can be hosted in external data sources, compliant with the regulation, while a fingerprint can be kept in public blockchains. As N.B Truong et. al. suggest, in permissioned blockchain many compliance problems can be addressed by properly establishing the GDPR specific roles (Data Processors, Data Controllers, Data Subjects etc) and developing an external access management on top of the blockchain system, including an intermediation on any exposed API.

3.6. Insights

To summarize the information presented, the blockchain is an excellent starting point for establishing a communication topology that natively ensures the need for security and privacy of the transported data. However, the exposure in the public space brings a new set of attack methods, slightly different from the classic models. Even in private exposure, the risks of data manipulation are still present. However, the technology delivers sufficient protection out-of-the-box, reducing the effort of developing additional methods to complement the shortcomings. For the migration of security and confidentiality control over the transported data, in the premises of the delivered service, the cloud paradigm can be a better facilitator for the communication function.

The public blockchain is an excellent driver for the secure transfer of digital assets over the Internet and for establishing their ownership. However, the computational cost, the financial cost and the diversity of network attack types, compared to the usual ones, often make it infeasible.

The private blockchain solves an important part of the problems of the public blockchain. It reduces the computational cost by using light consensus protocols and addresses private contexts. However, it tends to break the boundary of data anonymity due to the fact that it addresses small and closed communities, and the membership is certified. Also, maintenance can be slightly complex for the collection of functions it offers over the involved infrastructure. The cloud thus becomes relevant in the context of elasticity and versatility.

4. Data Distribution & Orchestration

Another topic developed around data privacy is data orchestration and data distribution, a proposal that concerns and emphasizes matters like data locality and data mobility and how intelligent orchestration can commision and decommission information with the purpose to ensure availability, security and privacy. Thus, as previously mentioned, it is important to have means to constraint where sensitive information arrives, how information is processed and where and how one can ensure multi-tenancy in distributed systems through smart orchestration. Therefore I explore the possibility to merge different data governance systems like cloud control plane and scheduling mechanism with High Performance and Grid Computing management routines, with the aim to consolidate, in a unified solution, the best from all paradigms. Furthermore as part of data mobility, briefly presented through the eyeglass of data transmission and data exposure, I focus on validating the consistency of containerization engines in delivering data via containers. Therefore, on one side, the data processors are loaded in containers, able to run out-of-the-box on heterogeneous systems, near the data. On the other side, I validate how containers can preserve privacy using the classical isolation taxonomy versus a weaker isolation. A container is a method to perform virtualization with support from the host kernel. Thus, it stays above the kernel space sharing it with the host system. In a more pragmatic definition, a container can be seen as a multiplication of the user space. From a security perspective, escaping from a container implies a high risk in corrupting the entire hosting system, as the most critical part of it is shared among other containers and user space.

4.1. Research Questions

- How can I orchestrate both data and processors over large distributed systems?
- How data locality[4] can be ensured in critical infrastructures?

4.2. Contributions

- Stan, Ioan-Mihail, Daniel Rosner, and Ştefan-Dan Ciocîrlan. "Enforce a global security policy for user access to clustered container systems via user namespace sharing." *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2020.
- Stan, Ioan-Mihail, Ştefan-Dan Ciocîrlan and Răzvan Rughiniș. “UNDERSTANDING THE OPPORTUNITIES OF APPLYING KUBERNETES SCHEDULING CAPABILITIES IN HIGH PERFORMANCE COMPUTING” *Scientific bulletin. Series C: electrical engineering and computer science*

[4] move processing where data is located

4.3. Case Study - User namespace unification for better security and privacy

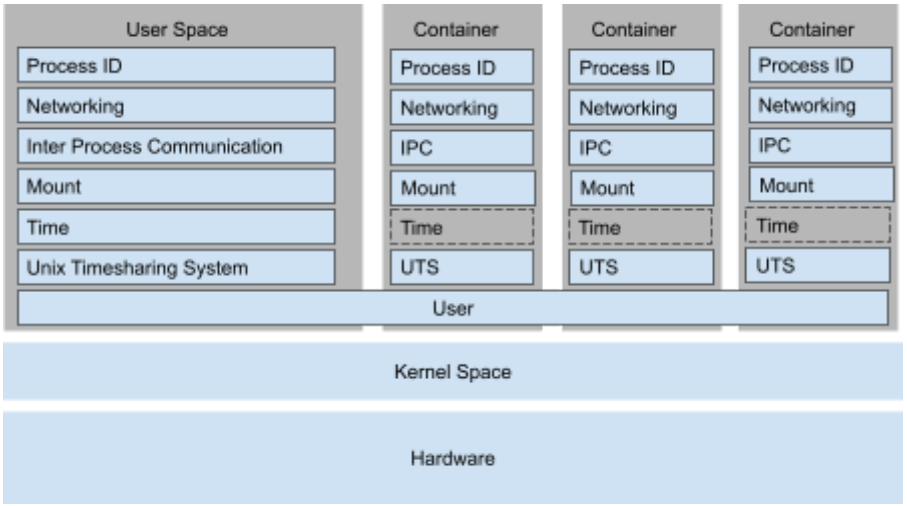


Figure 13- Conceptual isolation taxonomy for containers [135]

Modern, widely used, operating systems are mostly monolithic, meaning that between hardware (physical or emulated) there is an abstraction layer called Kernel space. This layer includes all drivers, schedulers and routines that interact optimally with the hardware. The kernel space offers services to the upper layer through system calls and enforces a protection ring model for the components managed. The superior layer, and the one that serves as an interface between the user's software and the operating system, is called user space.

The scope or the context of a user space is provided by 2 important features/services implemented in the Kernel (Linux Kernel): cgroups and kernel namespaces[136]. They can limit the scope of a user-space and provide an isolation taxonomy[136] (Figure 13 - containers isolation taxonomy with User namespace shared). When talking about containers, one can simply define a running container as being a separate instance of a user space - Figure 13. In this regard, by applying the same isolation taxonomy, the new instance of a user space will have its own objects and indexing mechanism and it will have mostly the same role as the main, native user space.

By switching the observation to the user namespace, this namespace covers all aspects of user management, including the user indexing system (UID). Therefore, a user with UID 1000 in the system user space is different from a user with UID 1000 defined in a container. A user policy applied at the level of the operating system may require adoption and reimplementation in the container.

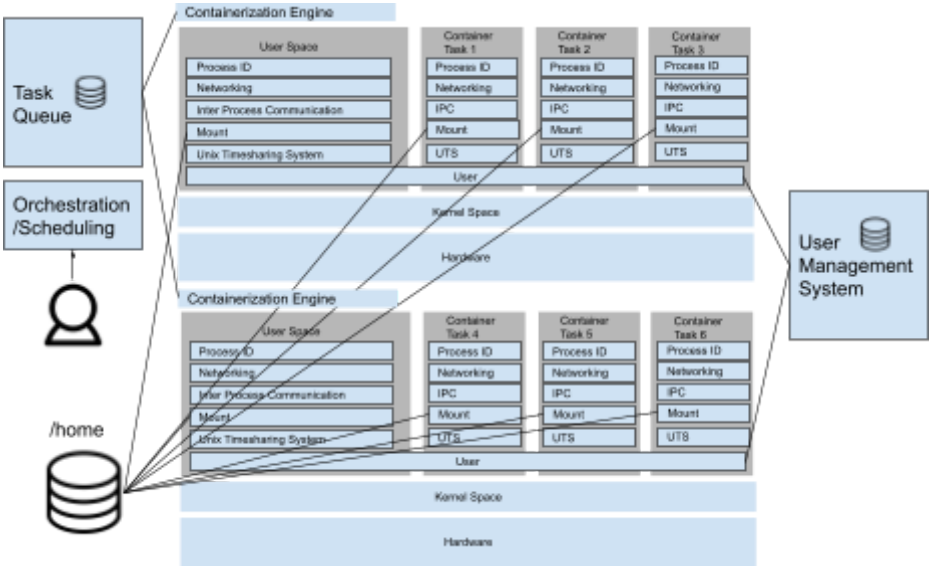


Figure 14- Concept architecture to support sharing a unique user namespace [135]

Access and user management is one of the most important primitives when designing a multi-tenant, multi-node secured system - Figure 14. Sometimes it can also involve special QoS Policies for access granularity, which can be implemented in a distributed manner or centralized. With the advancement of containerization, the security paradigm had to be extended, because, by spinning up a raw, conceptual container, one can impersonate the native user space ensemble. Therefore, a container user might be able to bypass some of the constraints that have been implemented at the system level, as they do not apply on other isolated user namespaces. In addition, if a process relies on the system access management to make some decisions, by running an instance of that process in a container, on a restricted physical node, one may be able to easily tamper the user indexing mechanism and to masquerade a malicious intention.

Conceptually speaking, this security aspect can have two simple resolutions, in relation with the capabilities that modern containerization engines are able to offer:

- limiting access to a list of essential people to the processing nodes and delegating the reinforcement of the user context and user policies to an external orchestrator or to the engine
- sharing the native system user namespace between host system and virtual enclaves (container)

With the former solution, a smart engine can apply a user remapping mechanism which will link users defined in a container to a predefined subset of system-defined non-privileged User IDs (UIDs). Therefore, all attempts to access system resources will be proxied through system user space and further executed without escalated privileges. Another method to overcome this challenge is to alter the default user context at bootstrap. Therefore, even if the original container image applies a specific user context, this will be changed when the container is launched. This approach is currently implemented by some container orchestrators (e.g., OpenShift OKD, Kubernetes) and it can be outsourced from the engine. However, sometimes this model may bring some compatibility issues. The most common problem is when processes/applications have been designed to run under the root account and a security policy forces the application to run as a non-root account. The application won't be able to run, and the containers will end up in an error state.

For the latter option, by simply attaching the system user namespace to all containers (Figure 13 and 14), all processes will inherit the same set of permissions and regulations from the user who requested the container execution. Despite the ideal model previously presented, in a pragmatic

approach, other namespaces may also be needed to partially merge into the system. They may need to provide additional services or objects to support the enforcement of a particular policy or setting (e.g., for user management, /etc/passwd and /etc/shadow files must be bind-mounted to containers, with read-only access).

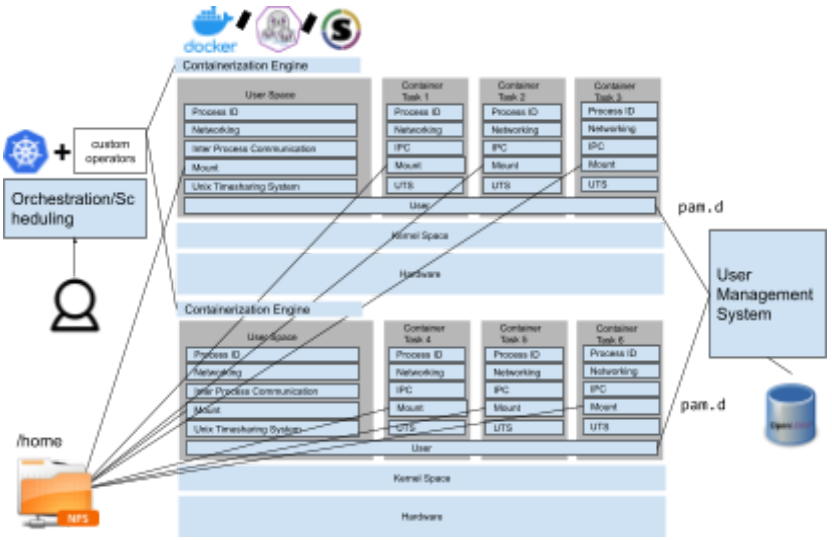


Figure 15 – Concept architectural design - implementation model [135]

In order to address the design (Figure 15) needs over the architecture proposed, I will analyze different containerization engines (Docker, Podman and Singularity) and I will reveal how their structure may sometimes embed a risk in bypassing the access boundaries by exposing escalation privileges opportunities. I will test how different containerization engines can be manipulated to run containers with unified user context and I will identify which are those services and artifacts that have to be shared between user space and container. The architecture proposed aims to provide critical infrastructures with the means to ensure a unified user policy context for properly maintaining the private assets and their processing in isolated, proprietary contexts. From the list proposed, Singularity acts better since its native isolation taxonomy proposes 1 layer isolation via the mount namespace. Docker, due to its client-server architecture, may present a risk since it may merge with the system root account, during the user namespace unification. Docker daemon runs as root, while the client can send requests from unprivileged users. Podman runs directly from the user context being designed as a client-only solution. Therefore, in the event of running containers with an unified user namespace, the mapping will fall back to the system user, which requested the container's creation.

In this case study [135], I have shown different containerization engines and their constraints in terms of the ease of merging the container user's namespace with the host user's namespace. I presented that, in the context of very complex QoS policies, tailored per individual, having a common user namespace shared between all virtual entities and the operating system, can increase the security of the entire ensemble. Moreover, extending this scenario to multi-node, multi-tenant topologies, I have identified a common clustered configuration in which such an approach can be essential - High-Performance Computing clusters.

The best containerization engine in regard to my case study is Singularity. This engine natively offers the function of sharing the user namespace. The solution is also compatible with artifacts developed for other containerization engines (e.g., Docker images). For Docker and Podman, similar results (user namespace merge) can be achieved through a sequence of manual steps or via an Orchestrator that can replicate and propagate user policies on each container, while deploying.

Finally, I propose a concept architecture and design for a multi-node, multi-tenant cluster that can support the unification of the user namespace [135]. I used OpenLDAP to outsource the user

management with posixAccounts, a Network File System to share any critical user artifact among all containers and Kubernetes to orchestrate the enforcement of the appropriate user context on containers.

4.4. Case Study - Adopting Kubernetes in High Performance computing

High Performance Computing is one of the most prolific concepts in Information Technology and one of the drivers of innovation. From the early stages of Covid-19 pandemics, companies and institutions brought together an incredible number of resources for running studies on the new virus. The Covid19 HPC Consortium shared freely around 6.4 million CPU cores, 603 Petaflops and 4.9k GPUs for running Covid-19 related projects. Still a rigid infrastructure, HPC has begun to adopt the container as a processing unit, inspired by the benefits they have brought to the cloud.

Another important game changer and innovator in the distributed systems market is Kubernetes, a container orchestrator, cloud-enabler, with a consistent adoption rate worldwide. As announced by CNCF (Linux Foundation), there are currently about 5.6 million developers who have adopted Kubernetes globally. With a well-developed control plan and an ecosystem built around technology, Kubernetes outperforms other competitors and becomes an industry standard.

As both distributed systems-based platforms focus their efforts on containers, the current case study proposes a classification of HPC-Kubernetes hybrid implementations and the way in which the important capabilities of both systems have merged. This taxonomy aims to analyze the implementations found in the literature, to group them.

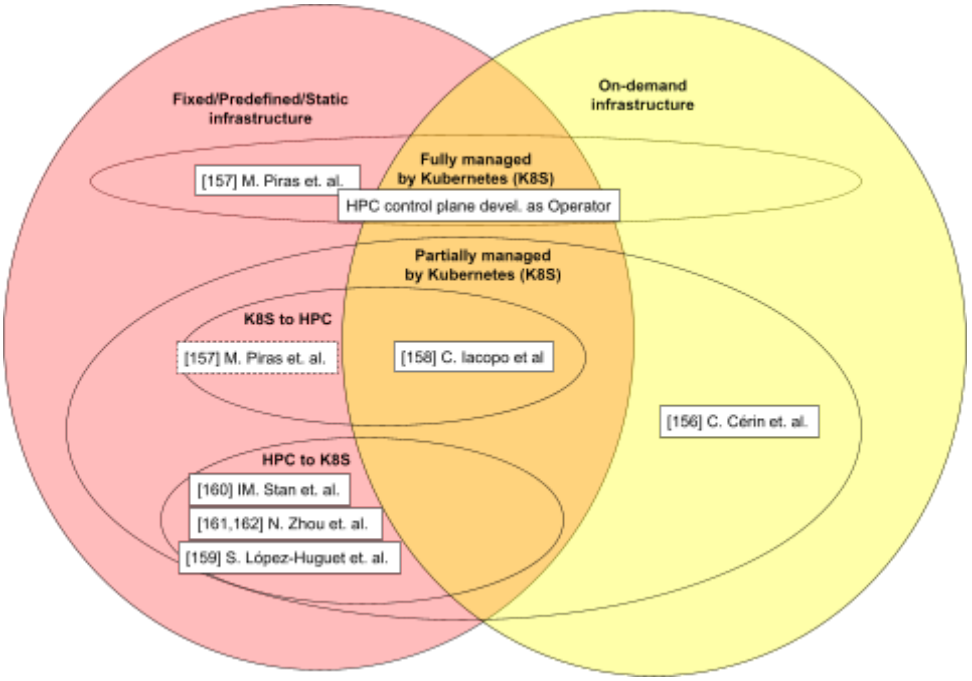


Figure 16 - Hybrid HPC-Kubernetes system classification model

The classification methodology comprises a simplified structure on three layers that aims to cover most implementations and to provide a better visibility of the integration methods used - Figure 16. The three layers taxonomy – fully-managed by Kubernetes, Kubernetes to HPC and HPC to Kubernetes - that define the form of communication between the control planes of the two distributed infrastructures is doubled by another dimensional axis, which represents the volatility of the hosting

infrastructure – fixed/pre-defined and provisioned on a demand basis. This observation perspective nuances the opportunities for cost optimization in terms of resource consumption.

The improvements suggested during the solutions analysis propose small but impactful changes in the architecture and design of the solutions found, especially in terms of replacing specific Kubernetes objects in certain use cases for optimizing the current workflows.

4.5. Insights

As could be seen in the case study presented in the section 4.3, by simplifying the classical container isolation taxonomy may ease the enforcement of global user policies. One one side, via the engine itself, if we're considering working on one-node topologies. On the other side, via smart orchestrators, which can manipulate multiple engines and enforce complex scheduling and constraints over containers provisioned. Therefore, such an approach shows a list of opportunities for setting boundaries over data and data processors in the context of critical distributed systems.

The case study briefly presented in section 4.4, proposes a classification taxonomy for solution found in recent scientific literature, as a way to emphasize the possibility to mix various distributed systems gouvernance models with the aim to address complex scenarios. Therefore such merges may enhance native scheduling mechanisms in order to address data locality problems or data distribution. In previous chapters, It has been proved how the cloud paradigm is quite versatile in scheduling capabilities, while batch-systems are highly constrained. Mixing both, may improve the capabilities to sustain modern privacy policies in regards to data and data processor scheduling.

5. Risk assessment

Cyber-attacks have intensified in recent times, as people tend to perform a larger spectrum of operations on the Internet. Similarly, malicious users have diversified their methods of attacking, proportionally to the new wave of Internet usage. To overcome the new security challenges, services running on the Internet may need to enforce their own protection mechanism, based on qualitative security analysis of the assets exposed. Running honeypots alongside production can become a standard, as by design, such a setup is able to capture behavioral attack information at runtime. Furthermore, as the adoption of the microservices paradigm has grown, such solutions tend to expand the attack surface, therefore, it is more relevant to validate the resilience of the assets exposed in case of a cyber attack and to generate consistent and organic risk assessments reports. Customization of protection policies, based on the attributes revealed by the risk assessment, can mitigate the risk of exposing confidential data.

5.1. Research Questions

- What are the opportunities to generate organic[5] security and privacy feedback/reports on assets/services and infrastructures exposing critical data?

5.2. Contributions

- Bontaș, Carol-Sebastian, Ioan-Mihail Stan and Răzvan Rughiniș. "Honeypot generator using software defined networks and recursively defined

[5] consistent, not incentivized

topologies." *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2022

- *Stan, Ioan-Mihail, Ştefan-Dan Ciocîrlan and Răzvan Rughiniş. "Deployment methodology and practises for running Hybrid HoneyPots together with the microservice- based production over Kubernetes"*[6]

5.3. Case Study - Honeypot Generator for randomizing deployment patterns

Running Honeypots is no longer an emerging trend but a necessity due to the increasing number of exploits. Such setups aim to provide a consistent view on the techniques cyber attackers use to obtain unauthorized access to private systems. Therefore, deploying honeypots may provide the means to understand and prevent 0-day vulnerabilities, complex exploiting techniques, malicious behavior between the clients of a platform and so on. A general problem of honeypots is that they may embed in their structure the developers fingerprint, therefore, successive deployments of a similar infrastructure within the premises of two distinct organizations may reveal the purpose of the infrastructure. One cyber criminal, attacking both setups, may see comparable patterns that can suggest it has been trapped in order to be observed and analyzed.

The recipes for deploying such complex infrastructures alongside the critical applications that must be protected are usually closed source. Governmental institutions or businesses tend to maintain their assets closed from the public, employing specialists in order to furnish defending techniques and configuration based on the industry standards. However, in the current context, attackers find refined methods to bypass the standards, every day. Therefore, the industry is in an unfair race with cyber criminals to understand the challenges and improve the security and privacy conventions in order to be offered to the general public.

Considering both challenges, my proposal is to deliver an open source solution for generating high-interaction honeypots that can serve both as a development framework and as a basis for the construction of complex research infrastructures, with an increased degree of randomization to reduce the problem of similarity in successive deployments. My approach tends towards the generation of consistent security and privacy reports of the services to be exposed to the public and the creation of dedicated policies to reduce the risks associated with a successful attack. Combined with the standards suggested by the industry, it could significantly increase the resistance of assets to malicious actions. The solution for generating honeypots, which can incorporate a production asset before release, is cost effective, being able to be run as a single system and simulating, with the help of containers, an entire physical infrastructure. The open source character of the high interaction honeypot generator offers exposure to consultants, specialists, developers who can contribute massively to the subsequent improvement of the solution. The developed system incorporates technological concepts such as Software Defined Networks, Recursively Defined Topologies and container orchestration. The proposed solution, which can constitute a framework for the development of honeypots with a large attack surface, introduces a mathematical formalism for the generation of recursive topologies with containers, a formal language for configuring the scalability parameters over the variables of the mathematical formalism, an algorithm for constructing topologies and a proprietary container orchestration methodology. At the same time, the case study proposes an extended, modern architecture that combines Software Defined Networks solutions with proprietary orchestration over containerization engines. Both facilitate the spin-up and deprovision of vulnerable services,

[6] Not published, summary and abstract accepted by DS Symposium Committee (http://doctorat.acs.pub.ro/ds-symposium-ro/)

dynamically, based on the attacker's behavior. The platform developed can be used for research purposes as part of the effort to counter novel attack methods or in the premises of IT organizations to expose services and simulate production. Before going further and presenting the concept architecture, it is mandatory to establish the premises and requirements [193]:

First, I set the focus on high-interaction honeypots for their complexity and relevance in critical sectors and research. Therefore they can help to establish the novel trends in cyber attacks and also reveal the tooling in use [169]. In regards to low-interaction honeypots, high-interaction honeypots require physical or virtual infrastructure since they act as legitimate production networks.

Secondly, I considered the dynamicity problem and the idea of reducing the development fingerprint in order to deliver legitimacy in a context of an attack. Therefore, the solution must respond to unpredictable scaling situations and must be able to react unassisted during the entire lifecycle of an attack or honeypot unfolding. The solution implements and adapts recursively defined topology heuristics such as FiConn[104] or DCell[170], usually used in datacenters construction for their ease on vertical scaling.

Thirdly, I learned from the construction techniques of cloud orchestrators and provided users with the capability to perform dynamic changes or establish the scaling capabilities from the very beginning, in order to allow the software to deal with the deployment operations automatically. Therefore vulnerable containerized services and network objects can be provisioned and deprovisioned based on observations. The smart heuristics implementing the control plane are brought together in order to build a proprietary container orchestrator and a framework for further development.

Last but not least, the solution must be resilient to unpredictable situations. Therefore it is important and mandatory to provide engineers with mechanisms to react quickly and to be able alter the network configuration or decommission vulnerable services in order to stop a complex attack. The framework proposed relies on Software Defined Networks [171] and Docker containers, which can be quickly and easily manipulated through proprietary management services.

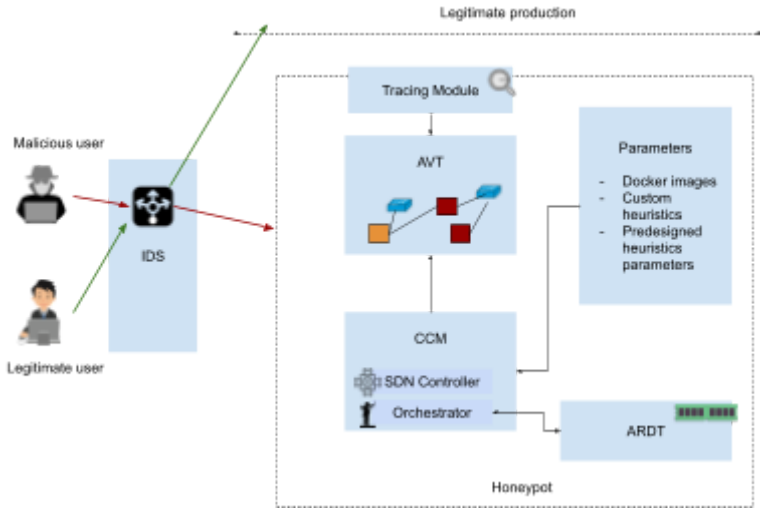


Figure 17 - Honeypot Generator Concept Architecture

The concept architecture (Figure 17) comprises six essential components [193]:

- Intrusion Detection System (IDS)
- Abstract Recursively Defined Topology module (ARDT)
- Tracing Module (MD)

- Central Command Module (CCM)
- Active Virtual Topology module (AVT)
- Parameter generator

The Intrusion Detection System is interposed in the frontend infrastructure and it is in charge to perform early detection of the user's intent[172][173][174][175] in regards to the services exposed. Therefore, if the user is malicious, the dynamic network infrastructure, delivered through SDN controllers and OpenVSwitch [176], will forward the potential attacker towards the honeypot. Legitimate users are certified by the same system and redirected to the production infrastructure[193].

The tracing module is in charge to analyze the current behavior of the malicious user while this one is trapped inside the honeypot. For simplicity, in the initial version of the system proposed, the Tracing Module is only able to detect and analyze privilege escalation attacks, where the attacker manages to exploit well purposed vulnerabilities in order to obtain a shell session towards the nearby containers[193].

The high interaction honeypot generator implements the lazy-evaluation strategy[193]. Therefore instead of deploying the entire container's infrastructure from the beginning, it provisions new vulnerable containers at runtime, during the lifecycle of an attack. In this regard, the system needs to always have knowledge about the location of the attacker and generate successive containers while neighbors are compromised. A simplified method to detect such occurrences is to capture change events in the shell history of a container. Therefore when an untouched container is detected as being manipulated from inside, it means that it has been exploited and the attacker is currently moving towards it. At this point, other containers are provisioned and part of the previous containers are decommissioned, based on a simple heuristic similar to minimax algorithm. The current active topology, the playground for the attacker, is figured in the architecture as AVT (Active Virtual Topology module). The logical structure (including provisioned, decommissioned and unprovisioned containers) is maintained in memory in the Abstract Recursive Defined Topology module. Thus, the control plane has a physical representation and a logical one, and knows all the time what needs to be provisioned, reprovisioned or decommissioned following the virtual image of the topology exposed and the attacker behavior[193].

The Central Command Module implements the orchestration function and gathers information or interacts with other modules in order to maintain the AVT in a consistent state. Therefore, it interprets the provisioning parameters and the heuristic selected, encapsulates the SDN Controller and the container orchestration function and gathers relevant information from ARDT. The Parameter Generator works as both a configuration management system and custom heuristic provider[193].

In addition, one of the aims of the solution proposed is that it can implement RDT-like algorithms such as DCell[170] and FiConn[104] to ensure scalability of the honeypot infrastructure and at the same time for predictability in resource management. Therefore, part of the research effort spent was concentrated on finding correlations between different heuristics. Through the configuration management system, one can define DCell or FiConn container topologies or other custom RDT dispatching heuristics [193]. The effort spent to understand the similarities between DCell and FiConn drived to defining a simplified mathematical formalism for RDT structures. Therefore, one can consider the following tuple (N, K, c, fc) [193] where

- N -> number of grade -1 hosts (hosts part of the basic structure, interconnected by switches)
- K -> a maximum grade to reach in the RDT topology, pre-established as a measure of predictability and resource management
- c -> a minimum grade from where the RDT structure imposes to have a fix number of links from c - graded structures towards other similar or higher grades structures

- fc -> the function that provides the number of links a c-graded structure needs to have with each grade structure

5.4. Case Study - Building Hybrid Honeypots Architectures over Kubernetes

In recent years, cyberattacks have seen a significant increase in numbers and success rates. A study led by Accenture showed that in 2021, the number of attacks per company during the year increased by 31% compared to 2020. From approximately 270 attempts of cyber-attacks, 29 of them succeeded. With the variety of attack methods and the growth of exploitable vulnerabilities, it is harder to scale generic security guidelines, pattern and threats detection models to cover the variety of services exposed to the Internet. However, such a process requires a broad understanding of the security aspects of the assets exposed. A method to obtain information, as quickly as possible, is to analyze attacker behavior on the premises of the service exposed to be consumed [181][182]. Running a honeypot that places the production assets in a controlled environment is a good way to generate organic security reports per item. A considerable challenge arises, when both production and the honeypot must be transparently displayed to both legitimate and malicious users. Thus, the entrance point towards the infrastructure must be shared between both infrastructure branches: production and honeypot. I call hybrid honeypot, critical infrastructure hosting both the production environment and a high-interaction honeypot in the premises of which attackers must be trapped while the ensemble still provides the production function.

Many of the Internet applications nowadays are constantly moving towards the microservices paradigm. The concept presented by Martin Fowler and James Luis proposes a way to break monoliths into small, independent, consumable services focused on delivering only one small function. Running together they can serve wider purposes. Those facilitators providing the means to build applications based on microservices architectures are the containerization engines (e.g., Docker) and container orchestrators (e.g., Kubernetes). The current study focuses on understanding various deployment methodologies and design patterns implemented in those technologies, with the aim to provide guidelines for generating hybrid honeypots for distributed containerized solutions.

This case study proposes a methodology for designing hybrid honeypots based on the microservices architecture with the aim to protect and isolate genuine assets. Architectural decisions have a pragmatic nuance and follow the idea of running distributed applications in containers over a Docker-Kubernetes ecosystem. The case study presents both architecture and design and introduces Kubernetes specific deployable objects, which can help to achieve a consistent isolation taxonomy. The evaluation of the methodology observes the mechanisms available in the Kubernetes ecosystem that ensure the multidimensional isolation of the production assets from those hosted in honeypots. The focus is on the segregation of the communication topology and on the resilience in the context of container-escape attacks.

In an overview, the methodology proposes 4 major stages:

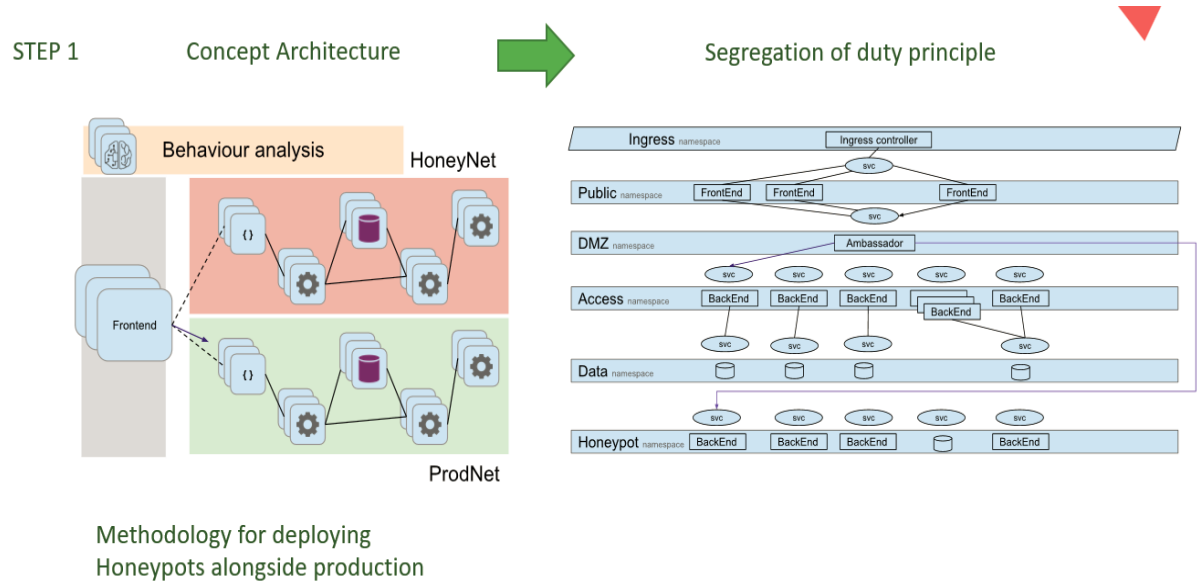
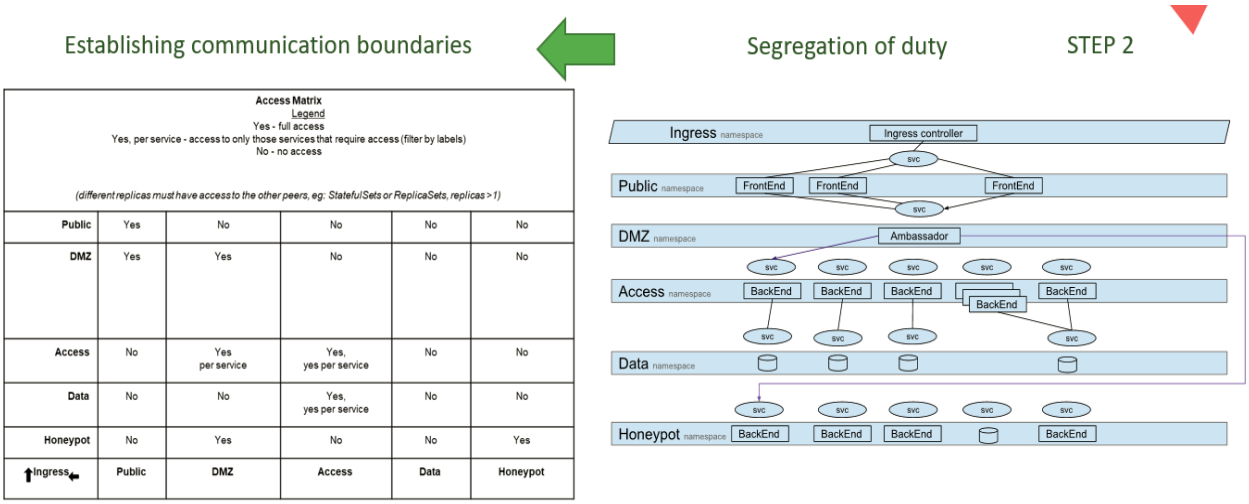


Figure 18 - Segregation of duty over Kubernetes

First stage proposes a split between microservices or containerized services in different namespaces (workspaces) with the aim to apply distinct protection policies (Figure 18) over the assets hosted. Therefore, one can segregate the public services/frontend services from backend and data layers, while a Demilitarized Zone is required to perform the forwarding function towards the production or the honeypot namespaces/workspaces.



Access Matrix	Public	DMZ	Access	Data	Honeypot
Public	Yes	No	No	No	No
DMZ	Yes	Yes	No	No	No
Access	No	Yes per service	Yes, yes per service	No	No
Data	No	No	Yes, yes per service	No	No
Honeypot	No	Yes	No	No	Yes
↑Ingress←	Public	DMZ	Access	Data	Honeypot

Figure 19 - Communication matrix

Second stage (Figure 19), proposes a communication and access matrix in order to define the communication boundaries for the containerized application, with respect for the architectural principles in microservices. Therefore, from a technical perspective, the stateful firewall function delivered by various network facilitators in Kubernetes, is consumed via Kubernetes native Network Policy objects.

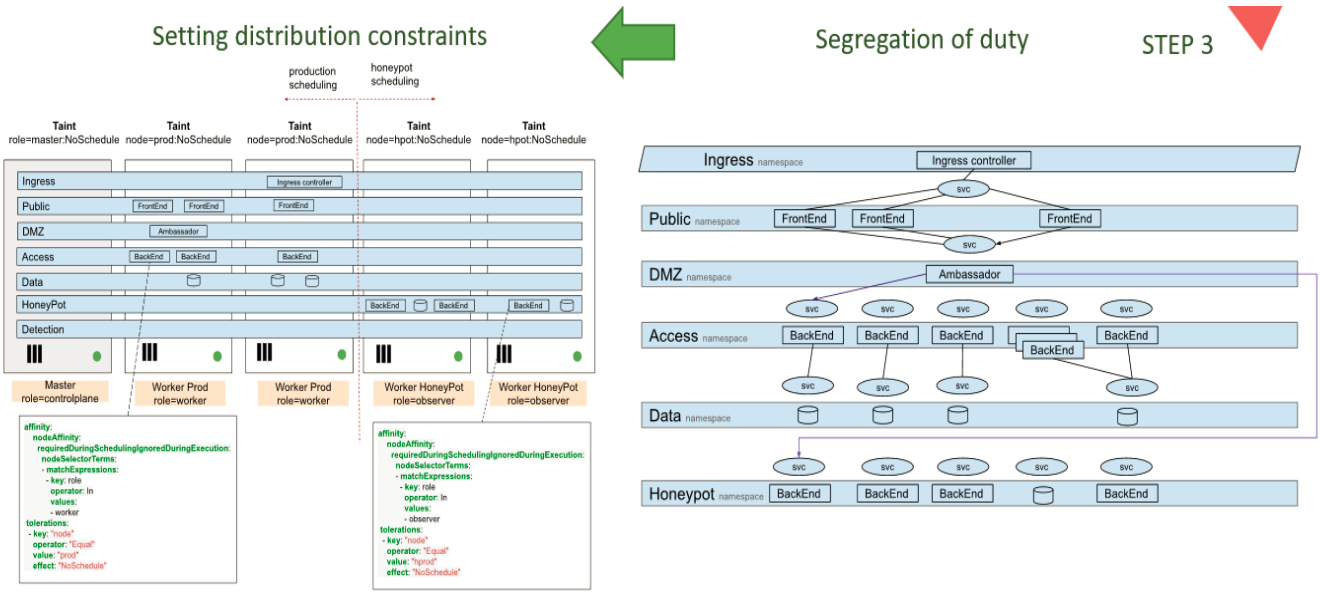


Figure 20 - Services distribution over Kuberentes clusters

Third stage (Figure 20) proposes how the segregation taxonomy can be distributed on Kubernetes clusters, in order to make possible the enforcement of distinct protection policies on the underlying infrastructure. Therefore, affinity function is used to alter the default behavior of the Kubernetes scheduler in order to spin-up specific application containers on specific nodes.

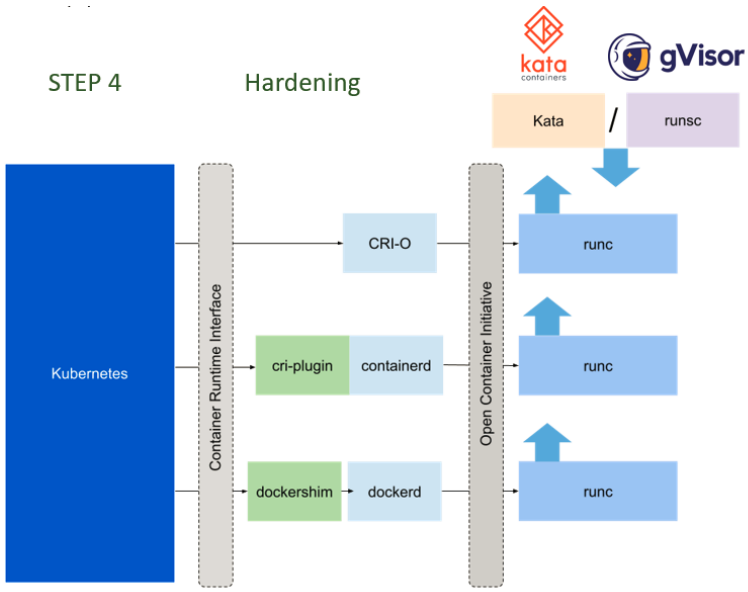


Figure 21 - Hardening in containerization engines

The fourth stage (Figure 21) considers the possibility to detach the native runtime of different containerization engines and replace it with more secure runtimes. Kata containers embed each container in small, lightweight virtual machines, to extend the system stack and to outsource the syscall API. gVisor with runsc, provides an admission controller over the existing kernel space, in order to filter any possible malicious syscall, originated from inside a container.

5.5. Insights

One relevant aspect when deploying critical assets is to understand the security risks that may lead to data exposure and therefore that may corrupt the borders established by data privacy objectives. Therefore, building honeypots around assets for early detection of major break-in points may constitute a real necessity especially in critical infrastructures. Therefore, one can expose early versions of the services in honeypots, prior to moving them to production, while in particular cases, both infrastructure branches can coexist.

In the case study presented in section 5.3, the solution proposed managed to establish a mix between two modern key concepts in dynamic service provisioning and scalable topologies: Software Defined Network and Recursively Defined Topologies. Through the platform provided, designed as a framework, researchers and developers can further generate high-interaction honeypots with the aim to counter 0-day exploits and to understand the novel cyber attacking techniques. Through its versatility and the fact that it is designed as a containerized infrastructure over Docker, it's easier to implement variable scenarios and expose one node topologies outside for validation.

In the case study presented in section 5.4, the work provides a methodology that constitutes the cornerstone for the construction of hybrid honeypots architectures over microservice applications. The bottom-up approach absorbs design models and methods from technologies like Kubernetes and Docker while both technologies are also providing the underlying infrastructure for the case study conducted. The methodology presents two main architectural views: the communication topology and the security and hardening stack deployed over the hosting nodes. Also, it shows how honeypots can be integrated into the production ecosystem, in a hybrid setup.

6. Conclusions

The work described in the current thesis provides relevant insights into the issue of privacy in complex and distributed infrastructures, with a focus on critical and expensive implementations. The thesis brings to the fore several models of governance over distributed systems, analyzing their ability to respond to current security and privacy issues. It also discussed how emerging technologies identified on the market, used in their original form, or adapted, can have a major impact on maintaining data privacy. Thus, I presented how blockchain can supplement the communication function, ensuring a consistent environment for anonymous cooperation. At the same time, the necessity of building honeypots around the production assets was considered, for a better understanding of the break-in points in the event of a public exposure.

The thesis addresses five major focal points in regard to analyzing data privacy matters: data accessibility, data anonymity, data transportation, data distribution & orchestration and risk assessment over the data exposed. I also present various case studies, developed around critical infrastructures such as government, financial or information technology sectors.

Bibliography

[1] Feldmann, Anja, et al. "The lockdown effect: Implications of the COVID-19 pandemic on internet traffic." *Proceedings of the ACM internet measurement conference*. 2020.

[2] Hijji, Mohammad, and Gulzar Alam. "A multivocal literature review on growing social engineering based cyber-attacks/threats during the COVID-19 pandemic: challenges and prospective solutions." *Ieee Access* 9 (2021): 7152-7169.

[3] Rinaldi, Steven M., James P. Peerenboom, and Terrence K. Kelly. "Identifying, understanding, and analyzing critical infrastructure interdependencies." *IEEE control systems magazine* 21.6 (2001): 11-25.
[4] Moteff, John, and Paul Parfomak. "Critical infrastructure and key assets: definition and identification." Library of Congress Washington DC Congressional Research Service, 2004.
[5] Forti, Alessandra, et al. "The fight against COVID-19: Running Folding@ Home simulations on ATLAS resources." *EPJ Web of Conferences*. Vol. 251. EDP Sciences, 2021.
[6] Foster, Ian, et al. "The grid2003 production grid: Principles and practice." *Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004.*. IEEE, 2004.
[7] Cesini, Daniele, et al. "The eXtreme-DataCloud project: data management services for the next generation distributed e-infrastructures." *2018 Conference Grid, Cloud & High Performance Computing in Science (ROLCG)*. IEEE, 2018.
[8] Koops, Bert-Jaap. "The trouble with European data protection law." *International data privacy law* 4.4 (2014): 250-261.
[9] Houser, Kimberly A., and W. Gregory Voss. "GDPR: The end of Google and Facebook or a new paradigm in data privacy." *Rich. JL & Tech.* 25 (2018): 1.
[10] Isaak, Jim, and Mina J. Hanna. "User data privacy: Facebook, Cambridge Analytica, and privacy protection." *Computer* 51.8 (2018): 56-59.
[11] Nam, Yeonghun, et al. "Global-scale GDPR Compliant Data Sharing System." *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, 2020.
[12] Politou, Eugenia, et al. "Backups and the right to be forgotten in the GDPR: An uneasy relationship." *Computer Law & Security Review* 34.6 (2018): 1247-1257.
[13] Xiao, Zhifeng, and Yang Xiao. "Security and privacy in cloud computing." IEEE communications surveys & tutorials 15.2 (2012): 843-859.
[14] Duan, Yucong, et al. "Everything as a service (XaaS) on the cloud: origins, current and future trends." *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 2015
[15] Höfer, C. N., and Georgios Karagiannis. "Cloud computing services: taxonomy and comparison." *Journal of Internet Services and Applications* 2.2 (2011): 81-94.
[16] Lu, Qinghua, et al. "uBaaS: A unified blockchain as a service platform." *Future Generation Computer Systems* 101 (2019): 564-575.
[17] Andersen, Michael P., Gabe Fierro, and David E. Culler. "Enabling synergy in iot: Platform to service and beyond." *Journal of Network and Computer Applications* 81 (2017): 96-110.
[18] Zhao, Feng, Chao Li, and Chun Feng Liu. "A cloud computing security solution based on fully homomorphic encryption." *16th international conference on advanced communication technology*. IEEE, 2014.
[19] de Castro, Leo, et al. "Does Fully Homomorphic Encryption Need Compute Acceleration?." *arXiv preprint arXiv:2112.06396* (2021).
[20] Merkel, Dirk. "Docker: lightweight linux containers for consistent development and deployment." *Linux j* 239.2 (2014): 2.
[21] De Lauretis, Lorenzo. "From monolithic architecture to microservices architecture." *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019.
[22] Jacob, Bart, et al. "Introduction to grid computing." *IBM redbooks* (2005): 3-6.
[23] Habib, Muhammad Asif, and Michael Thomas Krieger. "Security in Grid Computing." *Seminar aus Netzwerke und Sicherheit: Communication Infrastructure*. 2008.
[24] Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared." *2008 grid computing environments workshop*. Ieee, 2008.
[25] Dowd, Kevin, and Charles Severance. "High performance computing." (2010).
[26] Zhang, Di, et al. "RLScheduler: an automated HPC batch job scheduler using reinforcement learning." *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020.
[27] Mu'alem, Ahuva W., and Dror G. Feitelson. "Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling." *IEEE transactions on parallel and distributed systems* 12.6 (2001): 529-543.
[28] Courtès, Ludovic, and Ricardo Wurmus. "Reproducible and user-controlled software environments in HPC with Guix." *European Conference on Parallel Processing*. Springer, Cham, 2015.
[29] Sotiriadis, Stelios, et al. "From meta-computing to interoperable infrastructures: A review of meta-schedulers for HPC, grid and cloud." *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*. IEEE, 2012.
[30] Di Pierro, Massimo. "What is the blockchain?." *Computing in Science & Engineering* 19.5 (2017): 92-95.
[31] Sankar, Lakshmi Siva, M. Sindhu, and M. Sethumadhavan. "Survey of consensus protocols on blockchain applications." *2017 4th international conference on advanced computing and communication systems (ICACCS)*. IEEE, 2017.
[32] Rathod, Nidhee, and Dilip Motwani. "Security threats on blockchain and its countermeasures." *Int. Res. J. Eng. Technol* 5.11 (2018): 1636-1642.
[33] Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." *Decentralized Business Review* (2008): 21260.

[34] Gervais, Arthur, et al. "On the security and performance of proof of work blockchains." *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016.
[35] Tanwar, Sudeep, Karan Parekh, and Richard Evans. "Blockchain-based electronic healthcare record system for healthcare 4.0 applications." *Journal of Information Security and Applications* 50 (2020): 102407.
[36] Androulaki, Elli, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." *Proceedings of the thirteenth EuroSys conference*. 2018.
[37] Zhang, Jian, et al. "Deploying blockchain technology in the supply chain." *Computer security threats* (2019): 57.
[38] Ferdousi, Tanvir, Don Gruenbacher, and Caterina M. Scoglio. "A permissioned distributed ledger for the US beef cattle supply chain." *IEEE Access* 8 (2020): 154833-154847.
[39] Metcalfe, William. "Ethereum, smart contracts, DApps." *Blockchain and Crypt Currency* (2020): 77.
[40] Ayeni, O. A., B. K. Alese, and L. O. Omotosho. "Design and implementation of a medium interaction honeypot." *International Journal of Computer Applications* 975 (2013): 8887.
[41] Saputro, Elang Dwi, Yudha Purwanto, and Muhammad Faris Ruriawan. "Medium interaction honeypot infrastructure on the internet of things." *2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*. IEEE, 2021.
[42] Deogirikar, Jyoti, and Amarsinh Vidhate. "Security attacks in IoT: A survey." *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*. IEEE, 2017.
[43] Elmsheuser, Johannes, and Alessandro Di Girolamo. "Overview of the ATLAS distributed computing system." *EPJ Web of Conferences*. Vol. 214. EDP Sciences, 2019.
[44] Benjamin, D., A. Filipcic, and A. Klimentov. "ATLAS HPC Data Processing and Simulation."
[45] Fowler, Martin, and Matthew Foemmel. "Continuous integration." (2006).
[46] Ferrara, Pietro, and Fausto Spoto. "Static Analysis for GDPR Compliance." *ITASEC*. 2018.
[47] Lossent, Alexandre, A. Rodriguez Peon, and A. Wagner. "PaaS for web applications with OpenShift Origin." *Journal of Physics: Conference Series*. Vol. 898. No. 8. IOP Publishing, 2017.
[48] Ratis, Pavlos, Senior Site Reliability Engineer, and Red Hat. "Lessons Learned Using the Operator Pattern to Build a Kubernetes Platform." (2021).
[49] Schmeling, Benjamin, and Maximilian Dargatz. "Operations As Code with Kubernetes Operators and GitOps." *Kubernetes Native Development*. Apress, Berkeley, CA, 2022. 303-385.
[50] Mahboob, Jamal, and Joel Coffman. "A Kubernetes CI/CD Pipeline with Asylo as a Trusted Execution Environment Abstraction Framework." *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2021.
[51] Caban, William. "Architecting OpenShift Jenkins Pipelines." Architecting and Operating OpenShift Clusters. Apress, Berkeley, CA, 2019. 195-220.
[52] Ebert, Christof, et al. "DevOps." *Ieee Software* 33.3 (2016): 94-100.
[53] Kumar, Rakesh, and Rinkaj Goyal. "When Security Meets Velocity: Modeling Continuous Security for Cloud Applications using DevSecOps." *Innovative Data Communication Technologies and Application*. Springer, Singapore, 2021. 415-432.
[54] Beetz, Florian, and Simon Harrer. "GitOps: The Evolution of DevOps?." *IEEE Software* (2021).
[55] Barreiro, Fernando, et al. "The Future of Distributed Computing Systems in ATLAS: Boldly Venturing Beyond Grids." *EPJ Web of Conferences*. Vol. 214. EDP Sciences, 2019.
[56] Lukas, Wolfgang. "Fast simulation for ATLAS: Atlfast-II and ISF." *Journal of Physics: Conference Series*. Vol. 396. No. 2. IOP Publishing, 2012.
[57] Klimentov, A. A. "Exascale Data Processing in Heterogeneous Distributed Computing Infrastructure for Applications in High Energy Physics." *Physics of Particles and Nuclei* 51.6 (2020): 995-1068.
[58] Svirin, Pavlo, et al. "BigPanDA: PanDA Workload Management System and its Applications beyond ATLAS." EPJ Web of Conferences. Vol. 214. EDP Sciences, 2019.
[59] Korchuganova, Tatiana, et al. *The ATLAS BigPanDA Monitoring System Architecture*. No. ATL-SOFT-PROC-2018-019. ATL-COM-SOFT-2018-167, 2018.
[60] Stan, Ioan-Mihail, Siarhei Padolski, and Christopher Jon Lee. "Exploring the self-service model to visualize the results of the ATLAS Machine Learning analysis jobs in BigPanDA with OpenShift OKD3." *EPJ Web of Conferences*. Vol. 251. EDP Sciences, 2021.
[61] Beltre, Angel M., et al. "Enabling HPC workloads on cloud infrastructure using Kubernetes container orchestration mechanisms." *2019 IEEE/ACM International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC (CANOPIE-HPC)*. IEEE, 2019.
[62] Orzechowski, Michal, et al. "Transparent deployment of scientific workflows across clouds-kubernetes approach." *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, 2018.

[63] Bahadori, Kiyana, and Tullio Vardanega. "DevOps meets dynamic orchestration." *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, Cham, 2018.
[64] ATLAS Distributed Computing, https://twiki.cern.ch/twiki/bin/viewauth/ AtlasComputing/AtlasDistributedComputing (2021), accessed: 2021-01-28
[65] OKD - The Community Distribution of Kubernetes that powers Red Hat OpenShift, https://www.okd.io/ (2021), accessed: 2021-06-17
[66] Alekseev, A., et al. "ATLAS BigPanDA monitoring." *Journal of Physics: Conference Series*. Vol. 1085. No. 3. IOP Publishing, 2018.
[67] Padolski, S., et al. "Data visualization and representation in ATLAS BigPanDA monitoring." *Научная визуализация* 10.1 (2018): 69-76.
[68] Vukotic, Ilija, et al. *Atlas analytics and machine learning platforms*. No. ATL-SOFT-SLIDE-2018-417. ATL-COM-SOFT-2018-084, 2018.
[69] Radovic, Alexander, et al. "Machine learning at the energy and intensity frontiers of particle physics." *Nature* 560.7716 (2018): 41-48.
[70] Campana, Simone, and ATLAS collaboration. "ATLAS Distributed Computing in LHC Run2." *Journal of Physics: Conference Series*. Vol. 664. No. 3. IOP Publishing, 2015.
[71] Zaharia, Matei, et al. "Accelerating the machine learning lifecycle with MLflow." *IEEE Data Eng. Bull.* 41.4 (2018): 39-45.
[72] Serfon, Cedric, et al. "Rucio, the next-generation Data Management system in ATLAS." *Nuclear and particle physics proceedings* 273 (2016): 969-975.
[73] Burns, Brendan, and David Oppenheimer. "Design patterns for container-based distributed systems." *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*. 2016.
[74] Berghaus, Frank, et al. "Federating distributed storage for clouds in ATLAS." *Journal of Physics: Conference Series*. Vol. 1085. No. 3. IOP Publishing, 2018.
[75] Monreale, Anna, et al. "Movement data anonymity through generalization." *Trans. Data Priv.* 3.2 (2010): 91-121.
[76] Samarati, Pierangela, and Latanya Sweeney. "Generalizing data to provide anonymity when disclosing information." *PODS*. Vol. 98. No. 188. 1998.
[77] Zhang, Xuyun, et al. "An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud." *Journal of Computer and System Sciences* 79.5 (2013): 542-555
[78] Motwani, Rajeev, and Ying Xu. "Efficient algorithms for masking and finding quasi-identifiers." *Proceedings of the Conference on Very Large Data Bases (VLDB)*. 2007.
[79] Mansour, Huda O., et al. "Quasi-Identifier recognition algorithm for privacy preservation of cloud data based on risk reidentification." *Wireless Communications and Mobile Computing* 2021 (2021).
[80] Feng, Qi, et al. "A survey on privacy protection in blockchain system." *Journal of Network and Computer Applications* 126 (2019): 45-58.
[81] Gupta, Suyash, and Mohammad Sadoghi. "Blockchain transaction processing." *arXiv preprint arXiv:2107.11592* (2021).
[82] Al-Breiki, Hamda, et al. "Trustworthy blockchain oracles: review, comparison, and open research challenges." *IEEE Access* 8 (2020): 85675-85685.
[83] Zheng, Zibin, et al. "An overview on smart contracts: Challenges, advances and platforms." *Future Generation Computer Systems* 105 (2020): 475-491.
[84] Mense, Alexander, and Markus Flatscher. "Security vulnerabilities in ethereum smart contracts." *Proceedings of the 20th international conference on information integration and web-based applications & services*. 2018.
[85] Oliveira, Luis, et al. "To token or not to token: Tools for understanding blockchain tokens." (2018).
[86] Shirole, Mahesh, Maneesh Darisi, and Sunil Bhirud. "Cryptocurrency token: An overview." *IC-BCT 2019* (2020): 133-140.
[87] Davydov, Vyacheslav, et al. "Token standard for heterogeneous assets digitization into commodity." *Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications*. 2019.
[88] Victor, Friedhelm, and Bianca Katharina Lüders. "Measuring ethereum-based erc20 token networks." *International Conference on Financial Cryptography and Data Security*. Springer, Cham, 2019.
[89] Brünjes, Lars, and Murdoch J. Gabbay. "UTxO-vs account-based smart contract blockchain programming paradigms." *International Symposium on Leveraging Applications of Formal Methods*. Springer, Cham, 2020.
[90] Casale-Brunet, Simone, et al. "Networks of Ethereum non-fungible Tokens: a graph-based analysis of the ERC-721 ecosystem." *2021 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2021.
[91] Muthe, Koushik Bhargav, Khushboo Sharma, and Karthik Epperla Nagendra Sri. "A blockchain based decentralized computing and NFT infrastructure for game networks." *2020 Second International Conference on Blockchain Computing and Applications (BCCA)*. IEEE, 2020.

[92] Stan, Ioan-Mihail, Ilie-Constantin Barac, and Daniel Rosner. "Architecting a scalable e-election system using Blockchain technologies." *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2021.
[93] Cortier, Véronique, and Cyrille Wiedling. "A formal analysis of the Norwegian E-voting protocol." *International Conference on Principles of Security and Trust*. Springer, Berlin, Heidelberg, 2012.
[94] Alvarez, R. Michael, Thad E. Hall, and Alexander H. Trechsel. "Internet voting in comparative perspective: the case of Estonia." *PS: Political Science & Politics* 42.3 (2009): 497-505.
[95] Curran, Kevin. "E-Voting on the Blockchain." *The Journal of the British Blockchain Association* 1.2 (2018): 4451.
[96] Sajana, P., M. Sindhu, and M. Sethumadhavan. "On blockchain applications: hyperledger fabric and ethereum." *International Journal of Pure and Applied Mathematics* 118.18 (2018): 2965-2970.
[97] Adiputra, Cosmas Krisna, Rikard Hjort, and Hiroyuki Sato. "A proposal of blockchain-based electronic voting system." *2018 second world conference on smart trends in systems, security and sustainability (WorldS4)*. IEEE, 2018.
[98] Shi, Yichun, and Anil K. Jain. "DocFace+: ID document to selfie matching." *IEEE Transactions on Biometrics, Behavior, and Identity Science* 1.1 (2019): 56-67.
[99] Imerman, Michael B., and Frank J. Fabozzi. "Cashing in on innovation: a taxonomy of FinTech." *Journal of Asset Management* 21.3 (2020): 167-177.
[100] Fatrah, Aicha, et al. "Proof of concept blockchain-based voting system." *Proceedings of the 4th International Conference on Big Data and Internet of Things*. 2019.
[101] Thakkar, Parth, Senthil Nathan, and Balaji Viswanathan. "Performance benchmarking and optimizing hyperledger fabric blockchain platform." *2018 IEEE 26th international symposium on modeling, analysis, and simulation of computer and telecommunication systems (MASCOTS)*. IEEE, 2018.
[102] Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." *ACM SIGCOMM computer communication review* 38.4 (2008): 63-74.
[103] Lebiednik, Brian, Aman Mangal, and Niharika Tiwari. "A survey and evaluation of data center network topologies." *arXiv preprint arXiv:1605.01701* (2016).
[104] Li, Dan, et al. "FiConn: Using backup port for server interconnection in data centers." *IEEE INFOCOM 2009*. IEEE, 2009.
[105] Guo, Chuanxiong, et al. "BCube: a high performance, server-centric network architecture for modular data centers." *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 2009.
[106] Deshpande, Varun, Hakim Badis, and Laurent George. "Efficient topology control of blockchain peer to peer network based on SDN paradigm." *Peer-to-Peer Networking and Applications* 15.1 (2022): 267-289.
[107] Zhang, Jiboning. "Interaction design research based on large data rule mining and blockchain communication technology." *Soft Computing* 24.21 (2020): 16593-16604.
[108] Spasovski, Jason, and Peter Eklund. "Proof of stake blockchain: performance and scalability for groupware communications." *Proceedings of the 9th International Conference on Management of Digital EcoSystems*. 2017.
[109] Göbel, Johannes, and Anthony E. Krzesinski. "Increased block size and Bitcoin blockchain dynamics." *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2017.
[110] Vujičić, Dejan, Dijana Jagodić, and Siniša Ranđić. "Blockchain technology, bitcoin, and Ethereum: A brief overview." *2018 17th international symposium infoteh-jahorina (infoteh)*. IEEE, 2018.
[111] Reyna, Ana, et al. "On blockchain and its integration with IoT. Challenges and opportunities." *Future generation computer systems* 88 (2018): 173-190.
[112] Samaniego, Mayra, Uurtsaikh Jamsrandorj, and Ralph Deters. "Blockchain as a Service for IoT." *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 2016.
[113] Dannen, Chris. *Introducing Ethereum and solidity*. Vol. 1. Berkeley: Apress, 2017.
[114] Antonopoulos, Andreas M., and Gavin Wood. *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.
[115] Moubarak, Joanna, Eric Filiol, and Maroun Chamoun. "On blockchain security and relevant attacks." *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*. IEEE, 2018.
[116] Siddiqui, Shams Tabrez, et al. "Blockchain security threats, attacks and countermeasures." *Ambient Communications and Computer Systems*. Springer, Singapore, 2020. 51-62.
[117] Aggarwal, Shubhani, and Neeraj Kumar. "Attacks on blockchain." Advances in Computers. Vol. 121. Elsevier, 2021. 399-410.
[118] Huang, Dongyan, Xiaoli Ma, and Shengli Zhang. "Performance analysis of the raft consensus algorithm for private blockchains." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.1 (2019): 172-181.
[119] Popa, Alin Bogdan, Ioan Mihail Stan, and Răzvan Rughiniș. "Instant payment and latent transactions on the Ethereum Blockchain." *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2018.
[120] Wood, Ethereum. "A secure decentralised generalised transaction ledger, Ethereum Proj." *Yellow Pap* 151: 1.

[121] Vogelsteller, Fabian, and Vitalik Buterin. "ERC-20 token standard." *Ethereum Foundation (Stiftung Ethereum), Zug, Switzerland* (2015).
[122] Helliar, Christine V., et al. "Permissionless and permissioned blockchain diffusion." *International Journal of Information Management* 54 (2020): 102136.
[123] Liu, Manlu, Kean Wu, and Jennifer Jie Xu. "How will blockchain technology impact auditing and accounting: Permissionless versus permissioned blockchain." *Current Issues in auditing* 13.2 (2019): A19-A29.
[124] Cash, Michael, and Mostafa Bassiouni. "Two-tier permission-ed and permission-less blockchain for secure data sharing." *2018 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2018.
[125] Jang, Hyeji, Sung H. Han, and Ju Hwan Kim. "User perspectives on blockchain technology: user-centered evaluation and design strategies for dapps." *IEEE Access* 8 (2020): 226213-226223.
[126] Truong, Nguyen Binh, et al. "Gdpr-compliant personal data management: A blockchain-based solution." *IEEE Transactions on Information Forensics and Security* 15 (2019): 1746-1761.
[127] Schoo, Peter, et al. "Challenges for cloud networking security." *International Conference on Mobile Networks and Management*. Springer, Berlin, Heidelberg, 2010.
[128] Zhao, Zhifeng, Feng Hong, and Rongpeng Li. "SDN based VxLAN optimization in cloud computing networks." *IEEE Access* 5 (2017): 23312-23319.
[129] Benisi, Nazanin Zahed, Mehdi Aminian, and Bahman Javadi. "Blockchain-based decentralized storage networks: A survey." *Journal of Network and Computer Applications* 162 (2020): 102656.
[130] Barenji, Ali Vatankhah, et al. "Blockchain-based cloud manufacturing: Decentralization." *arXiv preprint arXiv:1901.10403* (2019).
[131] Ruan, Bowen, et al. "A performance study of containers in cloud environment." *Asia-Pacific Services Computing Conference*. Springer, Cham, 2016.
[132] Casalicchio, Emiliano. "Container orchestration: a survey." *Systems Modeling: Methodologies and Tools* (2019): 221-235.
[133] De Lucia, Michael J. *A survey on security isolation of virtualization, containers, and unikernels*. US Army Research Laboratory Aberdeen Proving Ground United States, 2017.
[134] Rosen, Rami. "Resource management: Linux kernel namespaces and cgroups." *Haifux, May* 186 (2013): 70.
[135] Stan, Ioan-Mihail, Daniel Rosner, and Ștefan-Dan Ciocîrlan. "Enforce a global security policy for user access to clustered container systems via user namespace sharing." *2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2020.
[136] Sun, Yuqiong, et al. "Security namespace: making linux security frameworks available to containers." *27th USENIX Security Symposium (USENIX Security 18)*. 2018.
[137] Dimou, Fani. "Automatic security hardening of Docker containers using Mandatory Access Control, specialized in defending isolation." (2019).
[138] Ghavamnia, Seyedhamed, et al. "Confine: Automated system call policy generation for container attack surface reduction." *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*. 2020.
[139] Biederman, Eric W., and Linux Networx. "Multiple instances of the global linux namespaces." *Proceedings of the Linux Symposium*. Vol. 1. No. 1. Citeseer, 2006.
[140] Jian, Zhiqiang, and Long Chen. "A defense method against docker escape attack." *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*. 2017.
[141] Lin, Xin, et al. "A measurement study on linux container security: Attacks and countermeasures." *Proceedings of the 34th Annual Computer Security Applications Conference*. 2018.
[142] Llopis, Pablo, et al. "Integrating HPC into an agile and cloud-focused environment at CERN." *EPJ Web of Conferences*. Vol. 214. EDP Sciences, 2019.
[143] Priedhorsky, Reid, and Tim Randles. "Charliecloud: Unprivileged containers for user-defined software stacks in hpc." *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 2017.
[144] Bui, Thanh. "Analysis of docker security." *arXiv preprint arXiv:1501.02967* (2015).
[145] Kurtzer, Gregory M., Vanessa Sochat, and Michael W. Bauer. "Singularity: Scientific containers for mobility of compute." *PloS one* 12.5 (2017): e0177459.
[146] Le, Emily, and David Paz. "Performance analysis of applications using singularity container on sdsc comet." *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. 2017. 1-4.
[147] Culic, Ioana-Maria, and Alexandru Radovici. "Development platform for building advanced Internet of Things systems." *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2017.
[148] Florea, Iulia, Laura Cristina Ruse, and Razvan Rughinis. "Challenges in security in Internet of Things." *2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2017.
[149] Rad, Babak Bashari, Harrison John Bhatti, and Mohammad Ahmadi. "An introduction to docker and analysis of its performance." *International Journal of Computer Science and Network Security (IJCSNS)* 17.3 (2017): 228

[150] Benedicic, Lucas, et al. "Sarus: Highly scalable Docker containers for HPC systems." *International Conference on High Performance Computing*. Springer, Cham, 2019.
[151]. Wang, Xingyu, Junzhao Du, and Hui Liu. "Performance and isolation analysis of RunC, gVisor and Kata Containers runtimes." *Cluster Computing* 25.2 (2022): 1497-1513.
[152]. Mavridis, Ilias, and Helen Karatza. "Orchestrated sandboxed containers, unikernels, and virtual machines for isolation‐enhanced multitenant workloads and serverless computing in cloud." *Concurrency and Computation: Practice and Experience* (2021): e6365.
[153]. Megino, Fernando Harald Barreiro, et al. "Using Kubernetes as an ATLAS computing site." *EPJ Web of Conferences*. Vol. 245. EDP Sciences, 2020.
[154]. Ferreira, Arnaldo Pereira, and Richard Sinnott. "A performance evaluation of containers running on managed kubernetes services." *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2019.
[155]. Wei-guo, Zhang, Ma Xi-lin, and Zhang Jin-zhong. "Research on kubernetes' resource scheduling scheme." *Proceedings of the 8th International Conference on Communication and Network Security*. 2018.
[156]. Cérin, Christophe, Nicolas Greneche, and Tarek Menouer. "Towards pervasive containerization of HPC job schedulers." *2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2020.
[157].Piras, Marco Enrico, et al. "Container orchestration on HPC clusters." *International Conference on High Performance Computing*. Springer, Cham, 2019.
[158]. Colonnelli, Iacopo, et al. "StreamFlow: cross-breeding cloud with HPC." *IEEE Transactions on Emerging Topics in Computing* 9.4 (2020): 1723-1737.
[159]. López-Huguet, Sergio, et al. "Seamlessly managing HPC workloads through Kubernetes." *International Conference on High Performance Computing*. Springer, Cham, 2020.
[160]. Stan, Ioan-Mihail et al. *Exploring the self-service model to visualize the results of the ATLAS Machine Learning analysis jobs in BigPanDA with OpenShift OKD3*. CERN, 2021
[161]. Zhou, Naweiluo, et al. "Container orchestration on HPC systems." *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. IEEE, 2020.
[162]. Zhou, Naweiluo, et al. "Container orchestration on HPC systems through Kubernetes." *Journal of Cloud Computing* 10.1 (2021): 1-14.
[163]. Dakic, Vedran, Mario Kovac, and Jasmin Redzepagic. "OPTIMIZING KUBERNETES PERFORMANCE, EFFICIENCY AND ENERGY FOOTPRINT IN HETEROGENOUS HPC ENVIRONMENTS." *Annals of DAAAM & Proceedings* 10.2 (2021).
[164] Dragoni, Nicola, et al. "Microservices: yesterday, today, and tomorrow." *Present and ulterior software engineering* (2017): 195-216.
[165] Larrucea, Xabier, et al. "Microservices." *IEEE Software* 35.3 (2018): 96-100.
[166] Stocker, Mirko, et al. "Interface quality patterns: Communicating and improving the quality of microservices Apis." *Proceedings of the 23rd European Conference on Pattern Languages of Programs*. 2018.
[167] Laigner, Rodrigo, et al. "Data management in microservices: State of the practice, challenges, and research directions." *arXiv preprint arXiv:2103.00170* (2021).
[168] Hemon, Aymeric, et al. "From agile to DevOps: Smart skills and collaborations." *Information Systems Frontiers* 22.4 (2020): 927-945.
[169] Alata, Eric, et al. "Lessons learned from the deployment of a high-interaction honeypot." *2006 Sixth European Dependable Computing Conference*. IEEE, 2006.
[170] Guo, Chuanxiong, et al. "Dcell: a scalable and fault-tolerant network structure for data centers." *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. 2008.
[171] Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE* 103.1 (2014): 14-76.
[172] Warrender, Christina, Stephanie Forrest, and Barak Pearlmutter. "Detecting intrusions using system calls: Alternative data models." *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*. IEEE, 1999.
[173] Forrest, Stephanie, et al. "A sense of self for unix processes." *Proceedings 1996 IEEE Symposium on Security and Privacy*. IEEE, 1996.
[174] Abed, Amr S., T. Charles Clancy, and David S. Levy. "Applying bag of system calls for anomalous behavior detection of applications in linux containers." *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015.
[175] Tunde-Onadele, Olufogorehan, et al. "A study on container vulnerability exploit detection." *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2019.
[176] Pfaff, Ben, et al. "The Design and Implementation of Open {vSwitch}." *12th USENIX symposium on networked systems design and implementation (NSDI 15)*. 2015.

[177] Mairh, Abhishek, et al. "Honeypot in network security: a survey." *Proceedings of the 2011 international conference on communication, computing & security*. 2011.
[178] Sokol, Pavol, Matej Zuzčák, and Tomáš Sochor. "Definition of attack in the context of low-level interaction server honeypots." *Computer Science and its Applications*. Springer, Berlin, Heidelberg, 2015. 499-504.
[179] Chovancová, Eva, and Norbert Ádám. "The Security of Heterogeneous Systems based on Cluster High-interaction Hybrid Honeypot." *2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES)*. IEEE, 2019.
[180] Wang, He, and Bin Wu. "SDN-based hybrid honeypot for attack capture." 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, 2019.
[181] Osman, Amr, et al. "Sandnet: towards high quality of deception in container-based microservice architectures." *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019.
[182] Kyriakou, Andronikos, and Nicolas Sklavos. "Container-based honeypot deployment for the analysis of malicious activity." *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018.
[183] Young, Ethan G., et al. "The True Cost of Containing: A {gVisor} Case Study." *11th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 19)*. 2019.
[184] Kumar, Rakesh, and B. Thangaraju. "Performance analysis between runc and kata container runtime." *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. IEEE, 2020.
[185] Sever, Dubravko, and Tonimir Kišasondi. "Efficiency and security of docker based honeypot systems." *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018.
[186] Viktorsson, William, Cristian Klein, and Johan Tordsson. "Security-performance trade-offs of kubernetes container runtimes." *2020 28th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2020.
[187] Combe, Theo, Antony Martin, and Roberto Di Pietro. "To docker or not to docker: A security perspective." *IEEE Cloud Computing* 3.5 (2016): 54-62.
[188] Reti, Daniel, and Norman Becker. "Escape the Fake: Introducing Simulated Container-Escapes for Honeypots." *arXiv preprint arXiv:2104.03651* (2021).
[189] Gomes, Jorge, et al. "Enabling rootless Linux Containers in multi-user environments: the udocker tool." *Computer Physics Communications* 232 (2018): 84-97.
[190] Tinney, Macdara. "Intrusion Detection for Kubernetes Based Cloud Deployments." (2020).
[191] Tsikerdekis, Michail, et al. "Approaches for preventing honeypot detection and compromise." *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, 2018.
[192] Gupta, Chakshu. *HoneyKube: designing a honeypot using microservices-based architecture*. MS thesis. University of Twente, 2021.
[193] Bontaș, Carol-Sebastian, Ioan-Mihail Stan and Răzvan Rughiniș. “Honeypot generator using software defined networks and recursively defined topologies.” *2022 21st RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2022