Non-Functional Certification of Modern Distributed Systems: From Cloud-Based Services to Machine Learning

Claudio A. Ardagna

Department of Computer Science,
Università degli Studi di Milano,
Milan, Italy

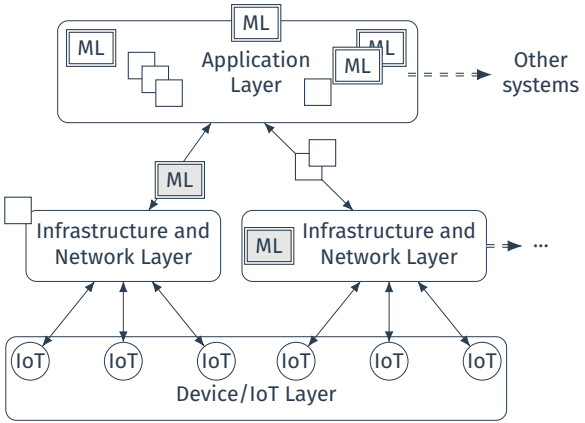
Università degli Studi di Padova
26th October 2023

Introduction

Scenario (1)

Modern distributed systems

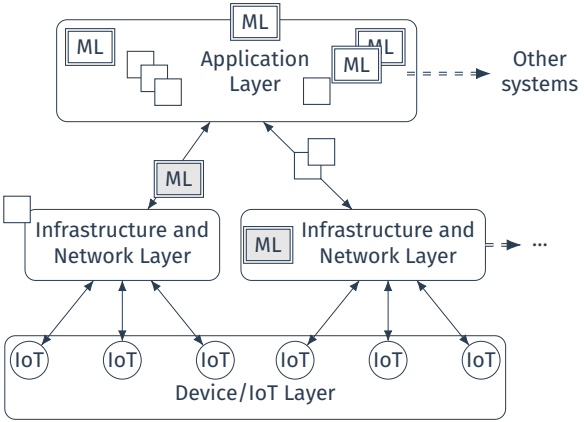
- confluence of cloud-edge-IoT
- multi-layer structure
- ML-based services and infrastructure
- dynamic, non-deterministic, and unpredictable behavior



Scenario (1)

Modern distributed systems

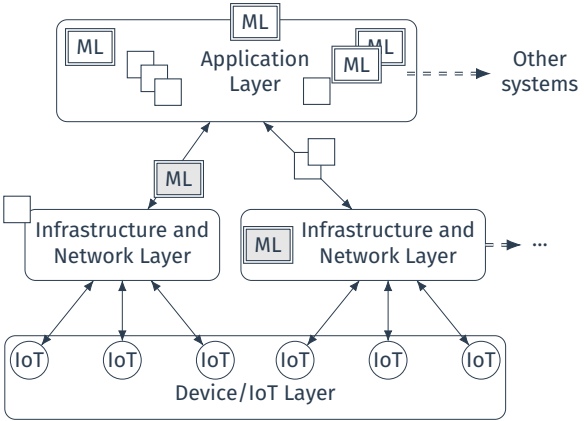
- impact of AI by 2030: $13 trillion (McKinsey)
- ≈15bln of connected devices in 2023: estimated to double by 2030 (Statista)
- economic impact of cloud-edge-IoT by 2025: $2.7–6.2 trillion (McKinsey)



Scenario (1)

Modern distributed systems

- increasing pervasiveness
- increasing risk for security, safety, and privacy
- lack of trustworthiness
  - full/partial lose of control on data/applications
  - lack of evidence about system operation and effectiveness



Scenario (2)

- Many users express little trust in the correctness and reliability of distributed systems they use
  - uncertainty on liability, consequences in case of failures, compliance to law
- Users' lack of trust has triggered an increasing public demand for improving the security of (mission and safety-critical) distributed systems
- A interdisciplinary research effort has been devoted to finding methods for creating security, safety, and dependability...

Scenario (2)

- Many users express little trust in the correctness and reliability of distributed systems they use
  - uncertainty on liability, consequences in case of failures, compliance to law
- Users' lack of trust has triggered an increasing public demand for improving the security of (mission and safety-critical) distributed systems
- A interdisciplinary research effort has been devoted to finding methods for creating security, safety, and dependability...

...but still we are far from a solution

Some Statistics

- Big enterprises and national critical infrastructures receive IT attacks every day
- Attacks are getting larger and more complex, while becoming increasingly inexpensive (ENISA ETL2023)
  - DDoS attacks peaking at 2.5Tbps (Cloudfare) and 46M request for second (Akamai)
  - RDoS are increasing mixing the danger of DDoS with the one of Ramsonware
  - Malware campaign increasingly involve IoT (“Mirai”, “Zerobot”, “MCCrash”)
- Cyberattacks as the fifth dimension of a war

Some Statistics

- Data breaches are inevitable (Microsoft)
- Since 2004, the total number of breached accounts is 16 billion (number accessed June 9, 2023), with 5,7 billion having a unique email address (Surfshark)
- Through 2023, government regulations requiring organizations to provide consumer privacy rights will cover 5 billion citizens and more than 70% of global GDP (Gartner)
- Data attacks to ML and AI are increasing (poisoning, adversarial attacks) (ENISA ETL2023)
- USD 4.35 million is the average total cost of a data breach that raises to USD 4.82 million when a critical infrastructure organization is targeted (IBM)

Some Statistics

- In 2022, it took an average of 277 days — about 9 months — to identify and contain a breach (IBM)
- The likelihood that a cybercrime entity is detected and prosecuted in the U.S. is estimated at around 0.05% (World Economic Forum)
- 54% of organizations have experienced a cyberattack in the last 12 months and 52% say there is an increase in cyberattacks compared to last year (Ponemon Institute – 2022)

Some Statistics

- Remote working is enlarging the boundaries of business infrastructures
- It takes organizations with a remote workforce 58 days longer to identify and contain the breach than office-based organizations (CyberTalk)
- Remote workers have caused a security breach in 20% of organizations during the pandemic (Malwarebytes)
- The advent of AI/ML provides a wealth of opportunities for both attackers and defenders

Cost of Known Attacks

3,62B$ in 2016

Cost of Known Attacks

Worldwide cybercrime costs will hit $10.5T annually by 2025

(Cybersecurity Ventures)

Opaque Systems Does Not Fit Security

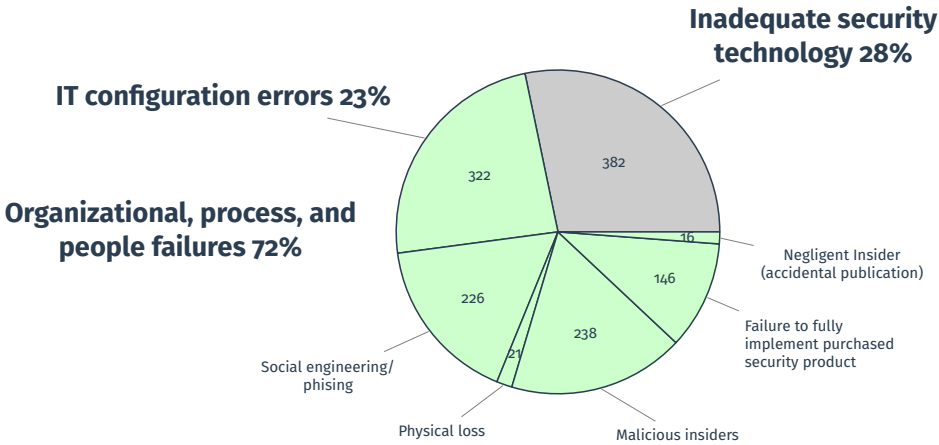
- Users distrust exacerbated in outsourced distributed systems (perceived lack of security)
  - service providers and customers lose (partially) control over the status of their data and applications
- Many solutions have been provided to improve the security of the cloud infrastructure
  - result: proliferation of ad hoc security solutions each targeting a very small part of the whole problem

Opaque Systems Does Not Fit Security

Need of increasing the system transparency and trustworthiness

- provide reliable and continuous evidence on the behavior of the distributed system and its (security) services
- manage the entire system lifecycle and its non-functional behavior
- involvement of customers and service providers in security management must be widened, increasing their trust

Security Breaches and Data Losses (1)



number of records lost (millions)

Security Breaches and Data Losses (2)

Main gaps:

- lack of appropriate compliance standards
- lack of continuous assurance and compliance assessment
- difficult integration between different application layers and heterogeneous security solutions

From Security to Assurance

Security: actively protects assets from threats and attacks

Assurance: gains justifiable confidence that infrastructure and/or applications will consistently demonstrate one or more security properties, and operationally behave as expected, despite failures and attacks

- wider notion than security, it collects and validates evidence supporting security properties according to audit, certification, and compliance to standards

C. A. Ardagna, R. Asal, E. Damiani, and Q. Vu. "From Security to Assurance in the Cloud: A Survey". In: *ACM Computing Surveys* 48.1 (2015)

Certification

Certification is an assurance procedure by which a third party gives written evidence that a product, process or service is in conformity with certain standards

- Certification techniques provide enough evidence that a system holds some non-functional properties and behaves correctly
- Certification has become widespread in the last 20 years and is also becoming important in today cloud environments
- A priori and run-time techniques

Certification History

Software Certification

- Fully manual
- Testing
- Focus on run-time target behavior
- Continuous (preliminary, later)

Service Certification

- Semi-automatic
- Testing
- Focus on run-time target behavior
- Continuous (preliminary)
- Certification-based system life cycle management

Cloud Certification

- Semi-automatic
- Multiple evidence
- Focus on run-time target behavior
- Continuous
- Certification-based system life cycle management
- Multi-layer

Modern Systems Certification

- Semi-automatic
- Multiple evidence
- Focus on run-time target behavior
- Continuous
- Certification-based system life cycle management
- Multi-layer
- Guidelines for ML

Certification in Europe

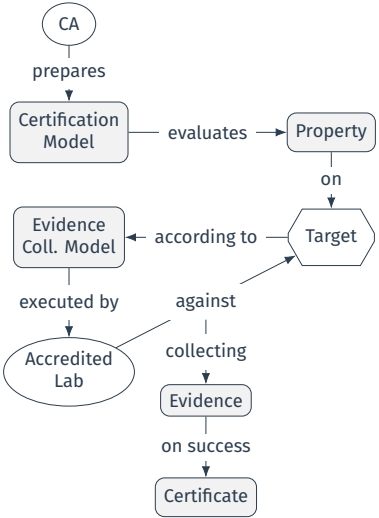
EU Cybersecurity Act

- new mandate for ENISA
  - EU Agency for Cybersecurity with permanent mandate, increased responsibilities and resources
  - setting up and maintain EU cybersecurity certification framework
  - operational cooperation at EU level for management of cyber incidents and large-scale attacks and crises
- European Cybersecurity Certification Framework
  - governance and rules for EU-wide certification of ICT products, processes, services
  - multiple schemes for different domains (*EUCC*, *EUCS*, *EU5G*)

Certification Scheme

Certification scheme details the certification process, according to

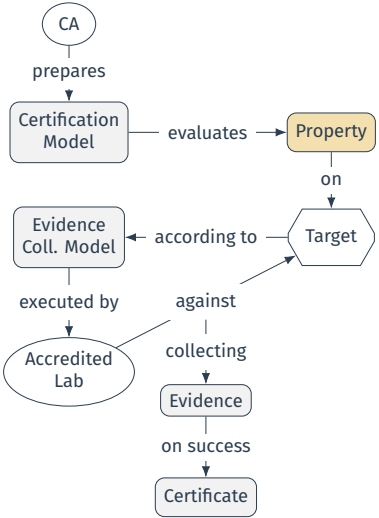
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

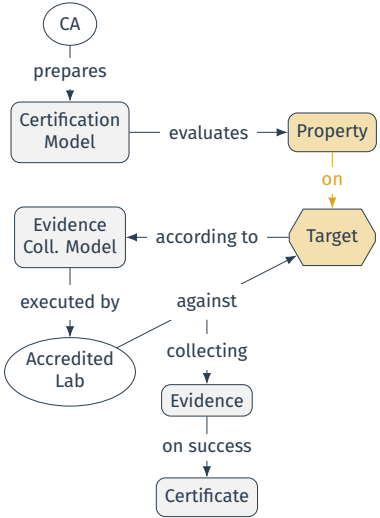
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

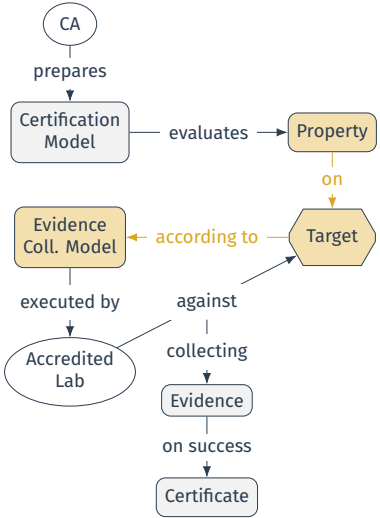
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

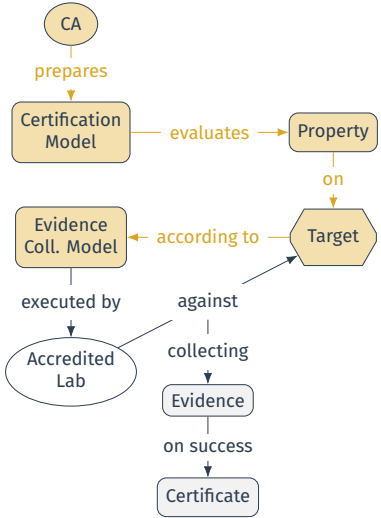
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

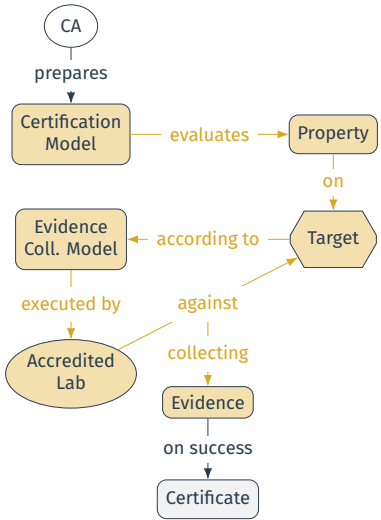
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

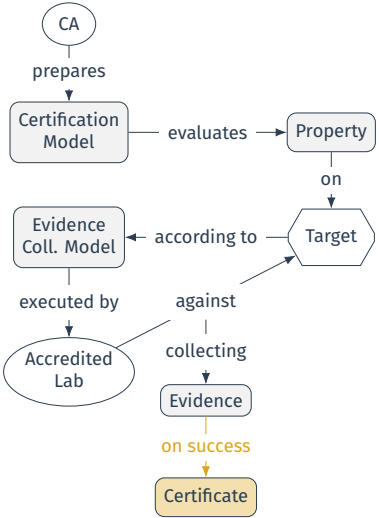
- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Scheme

Certification scheme details the certification process, according to

- non-functional property
- target of certification
- evidence collection model
- certification model
- evidence
- certificate



Certification Today (2)

Main challenges to face:

- **C1**: completeness
  - certificates awarded according to software artifacts only
    $\Longrightarrow$ multi-dimensional certification [5]
- **C2**: validity
- **C3**: integration
- **C4**: ML

C. A. Ardagna and N. Bena. "Non-Functional Certification of Modern Distributed Systems: A Research Manifesto". In: *Proc. of IEEE SSE 2023*. Chicago, IL, USA, July 2023

Certification Today (2)

Main challenges to face:

- **C1**: completeness
- **C2**: validity
  - lack of certification life cycle that keeps certificates updated with respect to the system changes
    $\Longrightarrow$ continuous certification [4]
- **C3**: integration
- **C4**: ML

C. A. Ardagna and N. Bena. "Non-Functional Certification of Modern Distributed Systems: A Research Manifesto". In: *Proc. of IEEE SSE 2023*. Chicago, IL, USA, July 2023

Certification Today (2)

Main challenges to face:

- **C1**: completeness
- **C2**: validity
- **C3**: integration
  - certification and development processes are disconnected and sequential
    $\Longrightarrow$ certification-driven development [16]
- **C4**: ML

C. A. Ardagna and N. Bena. "Non-Functional Certification of Modern Distributed Systems: A Research Manifesto". In: *Proc. of IEEE SSE 2023*. Chicago, IL, USA, July 2023

Certification Today (2)

Main challenges to face:

- **C1**: completeness
- **C2**: validity
- **C3**: integration
- **C4**: ML
  - ML requires a paradigm shift
    $\Longrightarrow$ ML certification [8]

C. A. Ardagna and N. Bena. "Non-Functional Certification of Modern Distributed Systems: A Research Manifesto". In: *Proc. of IEEE SSE 2023*. Chicago, IL, USA, July 2023

Multi-Dimensional Certification

C1: Completeness

Service	Dimension Artifacts D_{art}			Dimension Development D_{dev}					Dimension Evaluation D_{eval}	
	Repl.	Repl. Zones	HA Prot.	Prog. Lang.	Dev. Proc.	Type	State Mgmt.	Code Review	Trust. Contr.	When
s_1	3	3	Managed	Python	Waterfall	Monolith	Stateful	No	No	After
s_2	3	2	Custom	Java	Spiral	Microservice	Stateless	No	No	During
s_3	1	1	Custom	Java	Prototype	Monolith	Stateless	Yes	Yes	After
s_4	3	3	Managed	Rust	DevSecOps	Microservice	Stateless	Yes	Yes	During
s_5	2	3	Managed	Python	DevOps	Microservice	Stateless	No	Yes	After

Services s_1, s_4 are equivalent considering D_{art}

C1: Completeness

Service	Dimension Artifacts D_{art}			Dimension Development D_{dev}					Dimension Evaluation D_{eval}	
	Repl.	Repl. Zones	HA Prot.	Prog. Lang.	Dev. Proc.	Type	State Mgmt.	Code Review	Trust. Contr.	When
s_1	3	3	Managed	Python	Waterfall	Monolith	Stateful	No	No	After
s_2	3	2	Custom	Java	Spiral	Microservice	Stateless	No	No	During
s_3	1	1	Custom	Java	Prototype	Monolith	Stateless	Yes	Yes	After
s_4	3	3	Managed	Rust	DevSecOps	Microservice	Stateless	Yes	Yes	During
s_5	2	3	Managed	Python	DevOps	Microservice	Stateless	No	Yes	After

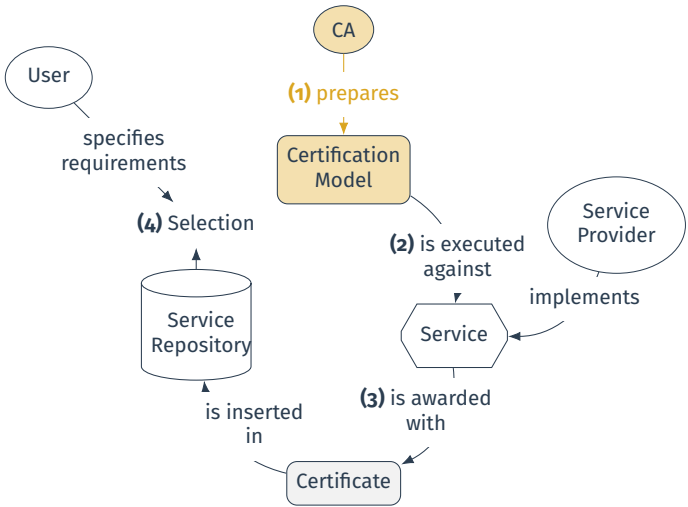
Services s_1, s_4 are equivalent considering D_{art}

$\Longrightarrow$ when considering additional aspects (D_{dev}, D_{eval}), s_4 is significantly better

Our Approach (1)

Multi-dimensional certification at the basis of system life cycle management

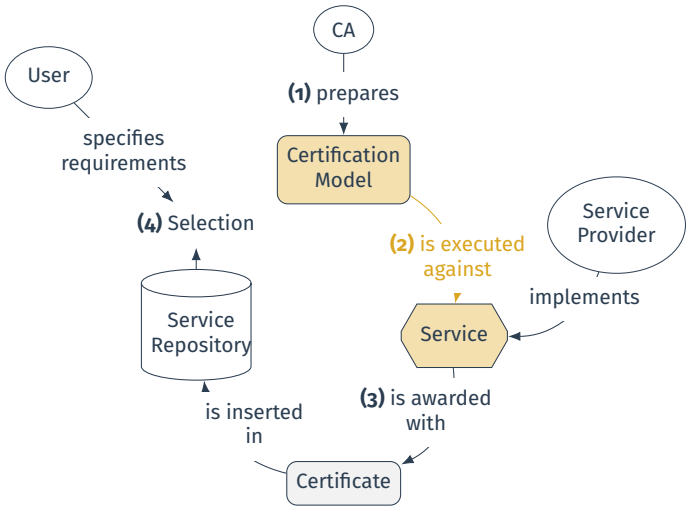
1. CA prepares the *certification model*
2. service is certified accordingly
3. and certificate is awarded
4. services are selected according to certificates



Our Approach (1)

Multi-dimensional certification at the basis of system life cycle management

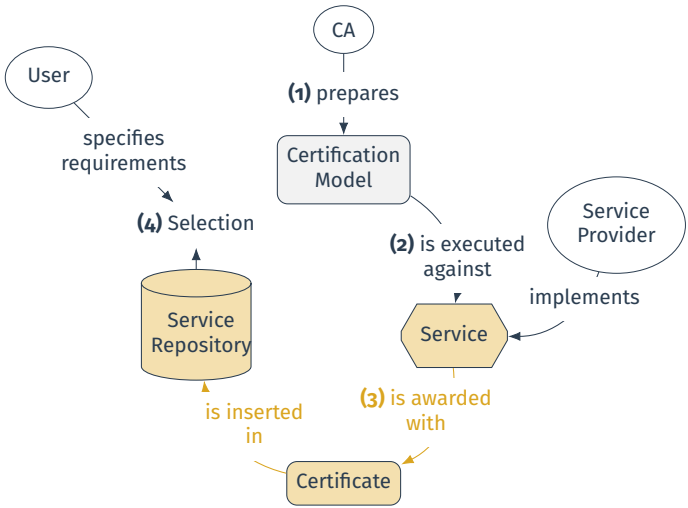
1. CA prepares the *certification model*
2. service is certified accordingly
3. and certificate is awarded
4. services are selected according to certificates



Our Approach (1)

Multi-dimensional certification at the basis of system life cycle management

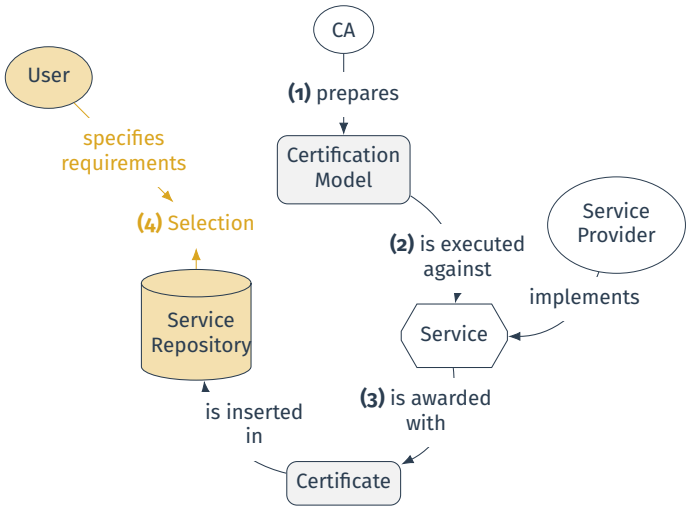
1. CA prepares the *certification model*
2. service is certified accordingly
3. and certificate is awarded
4. services are selected according to certificates



Our Approach (1)

Multi-dimensional certification at the basis of system life cycle management

1. CA prepares the *certification model*
2. service is certified accordingly
3. and certificate is awarded
4. services are selected according to certificates



From Non-Functional Property to Dimensions (1)

Attributes describing a non-functional property organized in *dimensions*

Dimension

A dimension D is a set $\{(a_1, v_1), \ldots, (a_n, v_n)\}$, where each attribute is a pair (a_i, v_i) with

- a_i the attribute name
- $v_i \in V_{a_i}$ the value for a_i, denoted as $a_i.v_i$

- each dimension describes a particular aspect of the non-functional property, e.g., software artifacts, development process

From Non-Functional Property to Dimensions (2)

A non-functional property is a set of dimensions

Non-Functional Property

A non-functional property is a pair $p=(\hat{p}, \{D_1, \ldots, D_n\})$, where

- $\hat{p}$ is an abstract property (i.e., the property name)
- D_i is a dimension

- no longer flat collection of attributes
- $\Longrightarrow$ finer-grained modeling enables finer-grained activities based on certificates

From Non-Functional Property to Dimensions (2)

A non-functional property is a set of dimensions

Considered dimensions

- D_{art}: software artifacts with attributes considered in the state of the art
  - *Repl.*
  - *Repl. Zones*
  - *HA Prot.*
- D_{dev}: development process
  - *Prog. Lang.*
  - *Dev. Proc.*
  - *Type*
  - *State Mgmt.*
  - *Code Review*
- D_{eval}: certification process
  - *Trust. Contr.*
  - *When*

From Non-Functional Property to Dimensions (2)

A non-functional property is a set of dimensions

Example

Property reliability: $p_{rel}=(\hat{p}_{rel}, \{D_{art}, D_{dev}, D_{eval}\})$

- D_{art}
  - *Repl.=3*
  - *Repl. Zones=3*
  - *HA Prot.=Managed*
- D_{dev}
  - *Prog. Lang.=Rust*
  - *Dev. Proc.=DevSecOps*
  - *Type=Microservice*
  - *State Mgmt.=Sateless*
  - *Code Review=Yes*
- D_{eval}
  - *Trust. Contr.=Yes*
  - *When=During*

Certification Model

The certification model details the activities to be performed to evaluate if a non-functional property holds on a service

- prepared by the CA
- executed by an accredited lab

- bound to a given non-functional property and service
- details the evidence to be collected and the rules regulating certificate award

Certification Model

The certification model details the activities to be performed to evaluate if a non-functional property holds on a service

Certification Model

A certification model is a tuple of the form $\mathcal{CM} = \langle p, ToC, \mathcal{E}, \mathcal{F} \rangle$, where

- p is the non-functional property
- ToC is the target of certification
- $\mathcal{E}$ is the evidence collection model
- $\mathcal{F}$ is the evaluation function determining the final outcome of the certification model execution

Target of Certification

The target of certification $ToC = \{\Theta_{D_1}, \ldots, \Theta_{D_n}\}$ models the service to be certified

$\Longrightarrow$ a set of non-functional mechanisms $\Theta_{D_i} = \{\theta_1, \ldots, \theta_m\}$ logically grouped according to dimensions

$\Longrightarrow$ mechanisms are the means by which the target supports a non-functional property

Example

$ToC = \{\Theta_{D_{art}}, \Theta_{D_{dev}}, \Theta_{D_{eval}}\}$, where

- $\Theta_{D_{art}} = \{$*Replica Manager*=Kubernetes$\}$
- $\Theta_{D_{dev}} = \{$*Pipeline*=File Content, *Source Code*=Rust, *Code Review Document*=File Content$\}$
- $\Theta_{D_{eval}} = \{$*Certification Framework*=Trusted-and-Continuous$\}$

Evidence

Evidence is collected during the certification process

- according to an *evidence collection model*, such as testing, monitoring, formal methods
- evidence is the result of the evidence collection model execution, providing the trust anchor of the awarded certificates
- evidence collection model $\mathcal{E}$ is a set of test cases $\{\{(\theta_1, t_1), \ldots, (\theta_n, t_n)\}\}$ insisting on specific mechanisms of the target
  - evidence collection model $\mathcal{E}_D$ instantiated on dimension D
  - test cases logically grouped according to dimensions they insist on
- evidence *ev* is collected successfully if all the test steps therein returns *Success*, denoted as $Succ(ev) = \top$

Example (Evidence Collection Model)

D	$\mathcal{E}_D$
D_{art}	$\{\{((\textit{Replica Manager}, \text{Kubernetes}), \text{Get-Orchestrator}), ((\textit{Replica Manager}, \text{Kubernetes}), \text{Check-Replicas}), ((\textit{Replica Manager}, \text{Kubernetes}), \text{Check-Zones})\}\}$
D_{dev}	$\{\{((\textit{Pipeline}, <\text{File Content}>), \text{Dry-Run})\}, \{((\textit{Source Code}, \text{Rust}), \text{Check-Code})\}, \{((\textit{Code Review Document}, <\text{File Content}>), \text{Check-Review-Document})\}\}$
D_{eval}	$\{\{((\textit{Certification Controls}, \text{Trusted-and-Continuous}), \text{Check-Document})\}\}$

Evaluation Function

The evaluation function determines the overall outcome of evidence collection

- a Boolean expression $\mathcal{F} = \mathcal{F}_{D_1} \wedge \ldots \wedge \mathcal{F}_{D_n}$
- composed of sub-evaluation functions $\mathcal{F}_{D_i} : \{ev\}_{D_i} \rightarrow \{\top, \bot\}$, each one determining the outcome of the certification model execution within a dimension D_i according to the collected evidence $\{ev\}_{D_i}$
- if successful, a certificate is awarded
- enables fine-grained rules for certificate award

Views

A view is a portion of the certification model, forming a single *space of evaluation* for a specific dimension

View

Let $\mathcal{CM}$ be a certification model and D a dimension in $\mathcal{CM}.p$. A view induced by D on $\mathcal{CM}$ is a tuple of the form $\mathcal{V}=\langle D, \Theta_D, \mathcal{E}_D, \mathcal{F}_D \rangle$, where

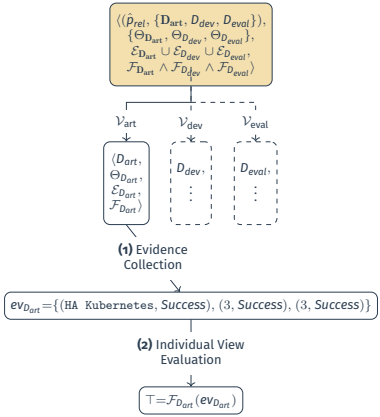
- D is the dimension
- Θ_D is the portion of the *ToC* corresponding to dimension D
- $\mathcal{E}_D$ is the evidence collection model of dimension D
- $\mathcal{F}_D$ is the portion of the evaluation function $\mathcal{F}$ returning the outcome of the certification model execution in the current view

We consider three views: $\mathcal{V}_{art}$, $\mathcal{V}_{dev}$, and $\mathcal{V}_{eval}$

Certification Model Execution

The execution of a certification model is a three-step process

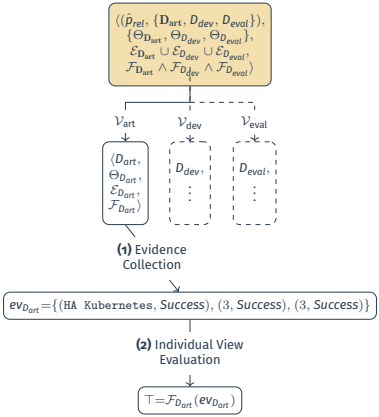
1. certification model view preparation
   1. A portion of the certification model $\mathcal{V}=\langle D, \Theta_D, \mathcal{E}_D, \mathcal{F}_D \rangle$, forming a single *space of evaluation* for a specific dimension ($\mathcal{V}_{art}$, $\mathcal{V}_{dev}$, and $\mathcal{V}_{eval}$)
2. certification model execution
   1. evidence collection
   2. individual view evaluation
3. certificate award



Certification Model Execution

The execution of a certification model is a three-step process

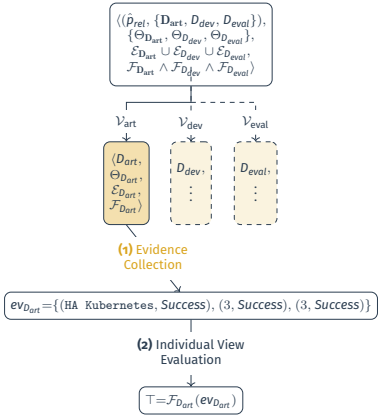
1. certification model view preparation
2. certification model execution, with the following steps executed in each view individually and independently
   1. evidence collection
   2. individual view evaluation
3. certificate award



Certification Model Execution

The execution of a certification model is a three-step process

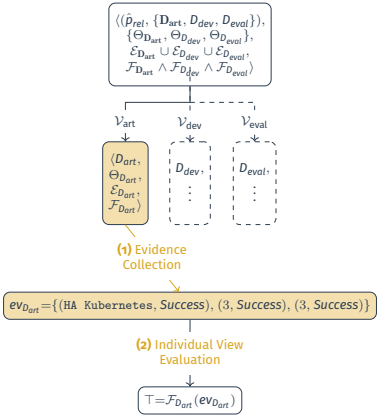
1. certification model view preparation
2. certification model execution, with the following steps executed in each view individually and independently
   1. evidence collection: evidence $\{ev\}_D$ is collected by executing the test cases in $\mathcal{V}.\mathcal{E}_D$ against the portion Θ_D of *ToC*
   2. individual view evaluation
3. certificate award



Certification Model Execution

The execution of a certification model is a three-step process

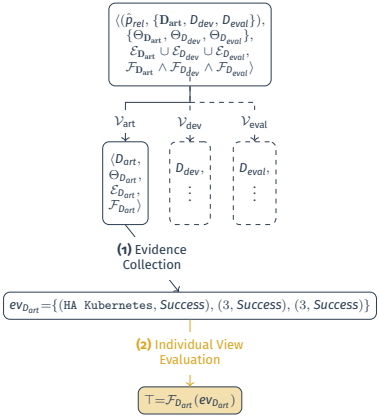
1. certification model view preparation
2. certification model execution, with the following steps executed in each view individually and independently
   1. evidence collection
   2. individual view evaluation: evaluation function $\mathcal{F}_D$ determines the result ($\top$ or $\bot$) of evidence collection
      - $\mathcal{F}_D$ is successful ($\top$) iff $\forall ev \in \{ev\}_D$, $Succ(ev) = \top$
      - $\mathcal{CM}.\mathcal{F}$ then aggregates individual results
3. certificate award



Certification Model Execution

The execution of a certification model is a three-step process

1. certification model view preparation
2. certification model execution
   1. evidence collection
   2. individual view evaluation
3. certificate award: if overall result is $\top$, a certificate is awarded



Certificate: Overview

The certificate includes three main components:

- the certification model
- the set of collected evidence
- a set of test metrics describing the evidence collection performance

Multi-Dimensional Certification: Service Selection

Certification-Based Service Selection

Certification opens the doors for subsequent activities and decisions based on trustworthy information (certificates)

$\Longrightarrow$ we consider service selection

Functionally-equivalent services are

1. matched against user's requirements (service filtering)
2. ranked according to their certificates (service ranking)

Dimension Lattice

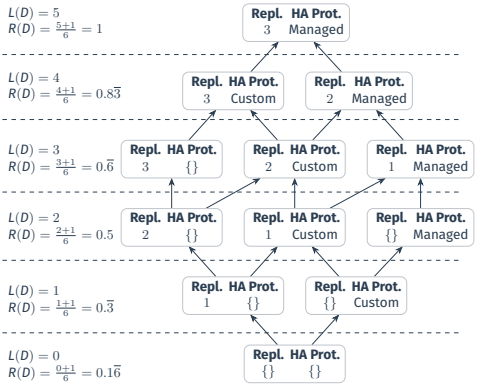
Each dimension induces a lattice, as the foundation for matching and ranking

$\Longrightarrow$ the lattice contains all the possible non-functional properties in the current dimension

A ranking function $R(D) = \frac{L(D)+1}{n}$ associates each element of the lattice with a numeric value

- D is a lattice element
- $L(D)$ returns the number of arcs of the minimum path from the least element to D in the corresponding Hasse diagram of the lattice
- $n = max(L(D)) + 1$

Example



Service Filtering

Service filtering filters certified services according to user's requirements in each dimension

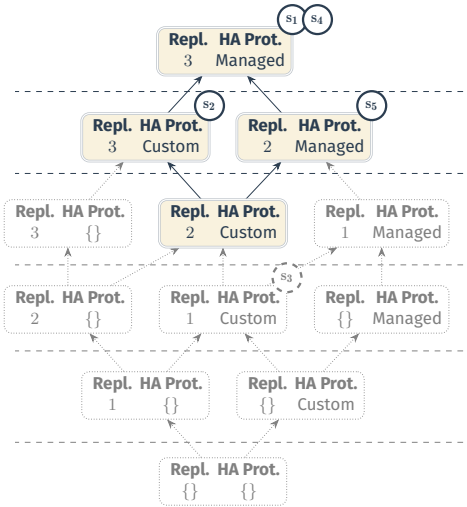
- A requirement is a set of the form $\{(glb, lub)_1, \ldots, (glb, lub)_n\}$

$\Longrightarrow$ service filtering selects compatible services whose certificate lay within the bound

Example

Input: user requirements

- *glb*: (*Repl*=2, *HA Prot.*=Custom)
- *lub*: (*Repl*=3, *HA Prot.*=Managed)



Service Ranking

Service ranking is modeled as a Multi-Criteria Decision-Making problem and solved according to the VIKOR algorithm

- finds the compromise solution, i.e., the closest to the ideal
  - among n alternative services $\{s_1, \ldots, s_n\}$
  - according to m conflicting criteria $\{D_1, \ldots, D_m\}$

Service Ranking

Service ranking is modeled as a Multi-Criteria Decision-Making problem and solved according to the VIKOR algorithm

Input

- set of compatible services $\{s_1, \ldots, s_n\}$
- corresponding certificates $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$
- vector W of weights reflecting the importance of each dimension for the user, s.t. $\sum_{i=0}^{|W|} W[i]=1$

Output

- ranking of $\{s_1, \ldots, s_n\}$

Multi-Dimensional Certification: Experiments

Quality Evaluation

Quality evaluation compares

- state of the art (considers dimension D_{art})
- global optimum (selects the services closest to the ideal)
- our approach

in terms of cumulative penalty

Captured Quality (1)

The penality of a service measures how much it diverges from the global optimum

$$P(s) = \sum_{i=0}^{n} W[i] \times \frac{max(D_i) - R(\mathcal{C}.\mathcal{CM}.p.D_i)}{max(D_i)}$$

$\Longrightarrow$ the sum of the normalized penalties contributed by each dimension

The quality of a service is defined as

$$QU(s) = 1 - \frac{P(s) - min(P)}{max(P) - min(P)}$$

Captured Quality (2)

Retrieve the 10 best services according our approach and state of the art and compute their average quality

Dim.	Our appr.	State of art
3	0.9178	0.755
6	0.9231	0.7166
9	0.924	0.6887
AVG	**0.9217**	**0.7201**

Captured Quality (2)

Retrieve the 10 best services according our approach and state of the art, and measure how many times (%) the best service according to the global optimum is also the best according to our approach and the state of the art

Dim.	Our appr.	State of art
3	72	16
6	70	16.667
9	76.6667	5
AVG	**72.889**	**12.556**

Overall Results

Experiments show that our approach

- captures 92% of the quality of the global optimum in all settings
- better approaches the global optimum-based ranking
- favors more *balanced* services

Continuous Certification

C2: Validity (1)

What makes a certificate *valid*?

A1) trust: certification model $\mathcal{CM}$ is created by a trusted CA, binding certificates to a chain of trust rooted at it

A2) correctness: $\mathcal{CM}$.*ToC* correctly represents the target system components; $\mathcal{CM}$.*p* correctly represents the property held by the system

A3) soundness: when evidence is successfully collected according to $\mathcal{CM}$, a certificate $\mathcal{C}$ is awarded proving that $\mathcal{CM}$.*ToC* supports $\mathcal{CM}$.*p*

C2: Validity (1)

What makes a certificate *valid*?

Valid Certificate

A certificate $\mathcal{C}$ is valid iff

- $\mathcal{CM}$ prepared by a trusted CA (A1) correctly represents the target system and its property (A2)
- $\mathcal{C}$ is released by successfully collecting sufficient evidence from the target system represented in $\mathcal{CM}$(A3)

C2: Validity (2)

Continuous certification: ensure that certificates remain valid along the target system evolution

- upon (any) changes, re-certification

Existing approaches fail to preserve this definition

- static target of certification: *ToC* is manually defined and assumed to remain the same across system changes
  $\Longrightarrow$ not true in modern systems

C2: Validity (2)

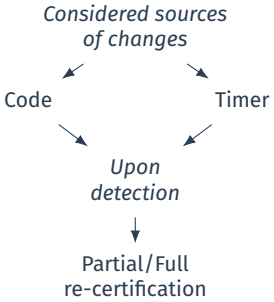
Continuous certification: ensure that certificates remain valid along the target system evolution

- upon (any) changes, re-certification

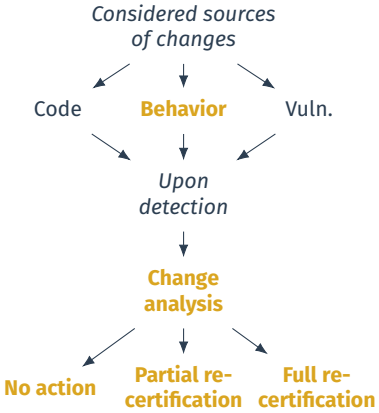
Existing approaches fail to preserve this definition

- lack of certification life cycle management
  - ineffective change detection: considered changes include code changes and pre-defined timers
    $\Longrightarrow$ code or timer do not imply a real change on *ToC* necessitating re-certification
    $\Longrightarrow$ behavioral changes unnoticed
  - inaccurate planning and inefficient adaptation: re-certification activities based on fixed rules that do not consider the cascading effects of changes

C2: Validity and Our Approach



(a) State of the Art



(b) Our Approach

Our Approach (1)

Certification Model

A certificate model $\mathcal{CM}_t$ at time t is a tuple of the form $\langle p, ToC, \{\{tc_i\}_{c_j}\}, \mathcal{CM}_{t-1}\rangle$ where

- p is the property
- ToC is the target
- $\{\{tc_i\}_{c_j}\}$ is the evidence collection model, with each test case tc_i insisting on a component $c_j \in ToC$
- $\mathcal{CM}_{t-1}$ is a reference to the certification model at time $t-1$ (if any)

Our Approach (2)

Certificate

A certificate $\mathcal{C}_t$ at time t is a tuple of the form $\langle \mathcal{CM}_t, \{ev\}_t, \mathcal{B}, \mathcal{C}_{t-1}, st\rangle$ where

- $\mathcal{CM}_t$ is the certification model
- $\{ev\}_t$ is the collected evidence, including
  - the new evidence $\{\overline{ev}\}$ collected at time t
  - the subset of evidence $\{ev\}_{t-1}$ in certificate $\mathcal{C}_{t-1}$ not superseded by evidence in $\{\overline{ev}\}$
- $\mathcal{C}_{t-1}$ is a reference to the certificate at time $t-1$ (if any)
- st is the certificate status retrieved using function $\texttt{state}\colon \mathcal{C}_t \to \{\text{Valid}, \text{Suspended}, \text{Superseded}, \text{Revoked}\}$

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
  $\longrightarrow$ $\mathcal{CM}$ and $\mathcal{C}$ still valid
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
  $\implies$ $\mathcal{CM}$ still represents the system
  $\implies$ new evidence to be collected to ensure $\mathcal{C}$ is still valid
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time *t*
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time *t* are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time *t*

  $\Longrightarrow$ some portions of $\mathcal{CM}$ no longer represent the system

  $\Longrightarrow$ $\mathcal{CM}$ needs to be updated and new evidence re-collected
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time *t*

Our Approach (3)

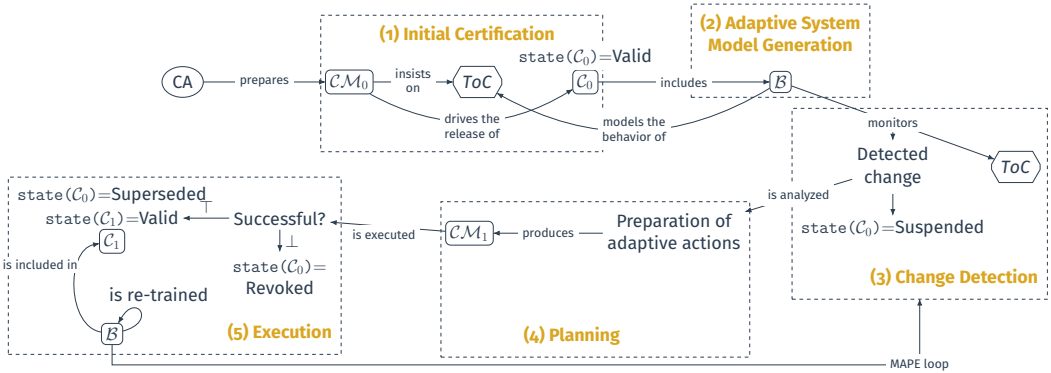
Four scenarios can happen during *ToC* evolution

- ***S0***: Certificate $\mathcal{C}_{t-1}$ is still valid: no changes observed at time t
- ***S1***: Certification model $\mathcal{CM}_{t-1}$ is still valid: changes at time t are minor
- ***S2***: Certification model $\mathcal{CM}_{t-1}$ needs revision: not-negligible changes at time t
- ***S3***: Certification model $\mathcal{CM}_{t-1}$ cannot be repaired: significant changes at time t

  $\Longrightarrow$ $\mathcal{CM}$ no longer represent the system

  $\Longrightarrow$ a new $\mathcal{CM}$ needs to be defined, and new evidence collected

Our Approach (4)



Reference Example

Example

Microservice managing applicable discounts in an e-commerce application

- service modeled as a set of components: $ToC = \{c_{\text{db}}, c_{\text{api}}, c_{\text{cross}}\}$
- property *performance*: $p_p = (\hat{p}_p, \{(\textit{lang}, \text{Java}), (\textit{max-time}, 10\text{ms}), (\textit{tolerance}, 1\text{ms})\})$, with $\hat{p}_p = \textit{Performance}$
- evidence collection model as a set of test cases insisting on components: $\{\{tc_1\}_{c_{\text{api}}}\}$, where $\{tc_1\}_{c_{\text{api}}}$ consists of one test case sending concurrent requests to the microservice and checking if the response time is compatible with p_p

Reference Example

Example

Microservice managing applicable discounts in an e-commerce application

At time t_0:

1. certification model $\mathcal{CM}_{t_0}$ is created: $\mathcal{CM}_0 = \langle p_p, ToC, \{\{tc_1\}_{c_{\text{api}}}\}, - \rangle$
2. evidence is collected and certificate $\mathcal{C}_{t_0}$ is released
3. $\mathcal{B}$ is created with initial data
4. the continuous certification process starts

Phase Change Detection

Phase change detection continuously monitor *ToC* to retrieve changes for later analysis. Considered changes:

- behavioral change anomalies in the system behavior , defined as $\Delta_b = (r, \{c_i\})$
  - $r \in \{\top, \bot\}$
  - $\{c_i\}$ is the set of components whose behavior changed
- code change: changes in the code base upon any releases
- vulnerability change: new vulnerability

Phase Change Detection

Phase change detection continuously monitor *ToC* to retrieve changes for later analysis. Considered changes:

- behavioral change anomalies in the system behavior
- code change: changes in the code base upon any releases, defined as $\Delta_c = (r, \{c_i\}, \mathcal{M})$
  - $r \in \{\top, \bot\}$
  - $\{c_i\}$ is the set of components whose code changed
  - $\mathcal{M} = \{M_i\}$ is a set of *complexity metrics*, each metric analyzes the source code at time $t-1$ and t to determine if the code change is *minor* or *major*
- vulnerability change: new vulnerability

Phase Change Detection

Phase change detection continuously monitor *ToC* to retrieve changes for later analysis. Considered changes:

- behavioral change anomalies in the system behavior
- code change: changes in the code base upon any releases
- vulnerability change: new vulnerability, defined as $\Delta_v = (r, \{(c, \texttt{vuln})\})$
  - $r \in \{\top, \bot\}$
  - $(c, \texttt{vuln})$ indicates the vulnerable component c and the corresponding vulnerability $\texttt{vuln}$

Phase Change Detection

Phase change detection continuously monitor *ToC* to retrieve changes for later analysis. Considered changes:

- behavioral change anomalies in the system behavior
- code change: changes in the code base upon any releases
- vulnerability change: new vulnerability

When at least one change at time t is detected:

- $\exists \Delta_i \in \{\Delta_b, \Delta_c, \Delta_v\}, \Delta_i.r = \top$
- certificate $\mathcal{C}_t$ is suspended: $\texttt{state}(\mathcal{C}_t) = \text{Suspended}$

Phase Change Detection

Phase change detection continuously monitor *ToC* to retrieve changes for later analysis

Example

At time t_1, a new version of the service is released, component c_{db} updated to improve its efficiency

- behavior of c_{db} changed
- behavior of c_{api} changed as cascading effect

Output: $\langle \Delta_{\text{b}}{=}(\top, \{c_{\text{api}}, c_{\text{db}}\}), \Delta_{\text{c}}{=}(\top, \{c_{\text{db}}\}, \{1\}), \Delta_{\text{v}}{=}(\bot, \emptyset) \rangle$, where

- $\{1\}$ is a metric value denoting the extent of the change on c_{db}
- change detected $\Longrightarrow$ $\texttt{state}(\mathcal{C}_{t-1}){=}\text{Suspended}$

Phase Planning (1)

Phase planning analyzes the detected changes at instant $t-1$ to retrieve the applicable scenario and adaptive action to obtain a valid certificate at instant t

Input: $\langle \Delta_b, \Delta_c, \Delta_v \rangle$

Output: $(\mathcal{CM}_t, \mathcal{T})$, where

- $\mathcal{CM}_t$ is a certification model correctly representing the system after changes at time t
- $\mathcal{T} \in \mathcal{CM}_t$ is a subset of test cases in $\mathcal{CM}_t$ to be later executed

Phase Planning (1)

Phase planning analyzes the detected changes at instant $t - 1$ to retrieve the applicable scenario and adaptive action to obtain a valid certificate at instant t

The output depends on the retrieved scenario, which is determined according to (set of) Boolean conditions over the changes

The first condition evaluated to $\top$ determine the applicable scenario

Phase Planning (2)

S#	Conditions	Output $(\mathcal{CM}_t, \{tc\})$
S0	– minor code change in non-critical, existing comp.	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1}$ – $\mathcal{T}$=$\emptyset$
S1	– behavioral change (environmental) – behavioral change (minor code change) – major code change without impact on behavior	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1}$ – $\mathcal{T}$=test cases on affected comp.
S2	– vulnerability discovered – behavioral change (major code change) in non-critical, existing comp.	– $\mathcal{CM}_t$=$\mathcal{CM}_{t-1} \cup \{tc\}_{\text{new}}$ – $\mathcal{T}$=test cases on affected comp.
S3	– behavioral change (environmental) in critical, existing comp. – major/minor code change in critical, existing comp. – code change adding a new comp.	– new $\mathcal{CM}_t$ – all test cases in $\mathcal{CM}_t$

Phase Execution

Phase execution executes the adaptive actions $\mathcal{CM}_t, \mathcal{T}$ to award a valid certificate $\mathcal{C}_t$

1. execution of test cases $\mathcal{T}$ to collect new evidence
2. system model re-training
   - $\mathcal{B}$ can thus detect behavioral changes in the new version of *ToC*
3. release of certificate $\mathcal{C}_t$
   - new certificate is valid: $\texttt{state}(\mathcal{C}_t) = \text{Valid}$
   - previous certificate is superseded: $\texttt{state}(\mathcal{C}_{t-1}) = \text{Superseded}$
   - previous certificate is revoked, otherwise: $\texttt{state}(\mathcal{C}_{t-1}) = \text{Revoked}$

Phase Execution

Phase execution executes the adaptive actions $\mathcal{CM}_t, \mathcal{T}$ to award a valid certificate $\mathcal{C}_t$

Example

Evidence is successfully collected from *ToC* according to $\mathcal{T}$

- new certificate $\mathcal{C}_{t_1}$ is released and is valid: $\texttt{state}(\mathcal{C}_{t_1})$=Valid, it contains
  - previous evidence in $\mathcal{C}_{t_0}$ not superseded by the new evidence
  - new evidence
- previous certificate is superseded: $\texttt{state}(\mathcal{C}_{t_0}) =$ Superseded

Continuous Certification: Experiments

Experiments

We evaluated

- the quality of $\mathcal{B}$ in detecting behavioral changes
- the quality of our scheme compared to a representative state-of-the-art-continuous certification scheme (SOTA)

$\mathcal{B}$ is implemented as a set of *isolation forest models*, each isolation forest is trained and detect anomalies on given system component

Target Systems

We used three distributed systems in literature

- *MS*, 38 microservices for streaming and reviewing movies
- *SN*, 36 microservices for a social network application
- *TT*, 41 microservices for train tickets management

For each distributed system we have the response time of the microservices in normal and anomalous conditions; we mapped

- each distributed system to a *ToC*
- each microservice to a component $c_i \in ToC$
- D_i denotes the dataset of distributed system i

Quality of Behavioral Changes Detection

Target system	ACC	PREC	REC
MS	0.8735	0.9971	0.8314
SN	0.6089	0.9794	0.4747
TT	0.9577	0.9969	0.9451
AVG	0.8133	0.9911	0.7504

- quality is relatively low in *SN* because normal and anomalous response times are often similar
- isolation forest models were not fine tuned

Comparative Evaluation (1)

D_{MS}, D_{SN}, D_{TT} provide information on behavioral changes only

$\Longrightarrow$ starting from them, we probabilistically generated datasets D'_{MS}, D'_{SN}, D'_{TT} including additional source changes

- we applied our scheme and SOTA on the datasets and measured their quality

Comparative Evaluation (2)

Procedure to generate an annotated dataset D'_i:

for $j = 1$ to 1000:

1. select if the data point to insert in the dataset represents a change (prob=0.4) or not (prob=0.6)
2. if change, select the change it represents between:
   - behavioral changes caused by environment (Δ_b)
   - code changes without impact on the behavior (Δ_c)
   - code changes with impact on the behavior (Δ_c with cascading)

Comparative Evaluation (2)

Procedure to generate an annotated dataset D'_i:

for $j = 1$ to 1000:

1. select if a critical component is involved (*critical*)
2. if code change (with or without impact on behavior):
   - select if the code change is minor (*minor*)
   - select if a component is involved ($n(comp)_{min}$ if change is minor, $n(comp)_{maj}$ otherwise)
3. if code change with impact on behavior
   - select the components whose behavior changed (uniform probability)
4. if environmental change:
   - select if a component is involved ($n(comp)_b$)

Comparative Evaluation (2)

Procedure to generate an annotated dataset D'_i:

for $j = 1$ to 1000:

- the probability that more than one component is involved in a change linearly decreases
- each row of the generated dataset contains
  - the response times of the components, selected from normal traces if the change does not involve behavior, anomalous traces otherwise
  - the annotations indicating the change, the extent, etc

Comparative Evaluation (3)

Experimental settings are divided in 4 groups *P1.*–P4.**

- *P1.**: uniform probabilities to Δ_b, Δ_c, Δ_c with cascading
- *P2.*–4.**: larger probability (0.5) to a change, uniform probability to the remaining ones (0.25)
  - *P2.**: larger probability to environmental change
  - *P3.**: larger probability to code change
  - *P4.**: larger probability to code with impact on behavior

Comparative Evaluation (3)

Experimental settings are divided in 4 groups *P1.*–P4.**

The remaining configurations determine the *profile* of the changes

- *P*.1*: major changes with critical impact (i.e., with impact on critical components) on a low number of components
- *P*.2*: average scenario balancing minor and major changes, critical and non-critical impact, on a medium number of components
- *P*.3*: minor changes with non-critical impact on a high number of components

Comparative Evaluation (4)

Name	Δ_b	Δ_c	Δ_c with cascad.	*critical*	*minor*	$n(comp)_b$	$n(comp)_{min}$	$n(comp)_{maj}$
P1.1	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.75	0.25	0.25	0.25	0.25
P1.2	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.5	0.5	0.5	0.5	0.5
P1.3	$0.3\overline{3}$	$0.3\overline{3}$	$0.3\overline{3}$	0.25	0.75	0.75	0.75	0.75
P2.1	0.5	0.25	0.25	0.75	0.25	0.25	0.25	0.25
P2.2	0.5	0.25	0.25	0.5	0.5	0.5	0.5	0.5
P2.3	0.5	0.25	0.25	0.25	0.75	0.75	0.75	0.75
P3.1	0.25	0.5	0.25	0.75	0.25	0.25	0.25	0.25
P3.2	0.25	0.5	0.25	0.5	0.5	0.5	0.5	0.5
P3.3	0.25	0.5	0.25	0.25	0.75	0.75	0.75	0.75
P4.1	0.25	0.25	0.5	0.75	0.25	0.25	0.25	0.25
P4.2	0.25	0.25	0.5	0.5	0.5	0.5	0.5	0.5
P4.3	0.25	0.25	0.5	0.25	0.75	0.75	0.75	0.75

Comparative Evaluation (5)

Experimental evaluation

1. for each initial dataset in D_{MS}, D_{SN}, D_{TT}, we generated 10 different annotated datasets D'_{MS}, D'_{SN}, D'_{TT}
2. we executed our scheme, detecting behavioral changes according to the isolation forest models
   - determine if there is a behavioral change according to the value of the data point
   - if annotated as code change, then it is a code change with impact on behavior, code change only otherwise
   - once the change is determined, it retrieves the applicable scenario

Comparative Evaluation (5)

Experimental evaluation

3. we executed a representative state-of-the-art continuous certification scheme
   - partial re-certification at minor code changes
   - full re-certification at major code changes, or any code changes affecting critical components
   - once the change is determined, it retrieves the applicable scenario

Comparative Evaluation (5)

Experimental evaluation

4. we then measured, for each scheme:
   - that is, ability of detecting all changes ($REC(changes)$)
   - ability to filter out scenarios where no adaptative action is required ($PREC(no\ action)$)
   - ability to detect all components involved ($REC(comp)$)
   - accuracy in retrieving the correct scenario ($ACC(S0)$–$ACC(S3)$)

Comparative Evaluation (5)

Experimental evaluation

- vulnerability-related changes are excluded since both schemes can detect it

Comparative Evaluation (6)

Name	REC(changes)		PREC(no action)		REC(comp)		ACC(scenarios)			
	Our	SOTA	Our	SOTA	Our	SOTA	ACC(S_0)	ACC(S_1)	ACC(S_2)	ACC(S_3)
P1.1	0.9948	0.6693	0.993	0.8179	0.8622	0.5727	0.8464	0.8716	0.9996	0.8622
P1.2	0.9894	0.6892	0.9882	0.8289	0.8769	0.5205	0.8397	0.8789	1	0.826
P1.3	0.9858	0.6741	0.9846	0.8242	0.8808	0.4215	0.8397	0.8456	0.9995	0.8402
P2.1	0.9958	0.6751	0.9946	0.8153	0.8633	0.5664	0.8445	0.8744	0.9998	0.8579
P2.2	0.9933	0.6738	0.9927	0.8203	0.8718	0.4968	0.844	0.8761	0.9998	0.8133
P2.3	0.9835	0.6645	0.9822	0.8218	0.8857	0.3874	0.835	0.8577	1	0.8072
P3.1	0.9948	0.6733	0.994	0.821	0.8676	0.5838	0.8469	0.9059	0.9998	0.8229
P3.2	0.9912	0.6761	0.9893	0.8184	0.8724	0.5129	0.8434	0.8621	1	0.8355
P3.3	0.9856	0.6726	0.9844	0.8157	0.8808	0.4037	0.8402	0.8549	0.9998	0.8172
P4.1	0.9949	0.6822	0.9947	0.8266	0.8707	0.5914	0.8439	0.8834	0.9998	0.8684
P4.2	0.9911	0.6821	0.9911	0.8373	0.8723	0.5325	0.8373	0.8404	0.9998	0.8802
P4.3	0.9875	0.6943	0.9858	0.8246	0.883	0.4238	0.8434	0.8494	0.9995	0.8391
AVG	**0.9906**	**0.6772**	**0.9895**	**0.8227**	**0.874**	**0.5011**	**0.842**	**0.8667**	**0.9998**	**0.8392**

ML Certification

Certification of Machine Learning

Certification of ML requires to

- define and evaluate non-functional properties with specific techniques [2]
- certify the ML model with multi-dimensional certification [8]
- certify the entire system (partially) driven by ML models
- and strengthen ML models [3]

Non-Functional Properties

- Transparency: explainability, interpretability
- Performance: the performance of a computation
- Privacy: ML models not storing information about their training set, nor such information can be inferred from the models' output
- Reliability: the ability of the software to operate without failures and to maintain a certain level of performance
- Safety: system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered
- Fairness: absence of discrimination, that is, prejudice against an individual or a group based on the value of specific features
- ...

M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero. "A Methodology for Non-Functional Property Evaluation of Machine Learning Models". In: *Proc. of MEDES 2020*. Abu Dhabi, UAE, Nov. 2020

Multi-Dimensional ML Certification

Adaptation of our multi-dimensional certification scheme for ML certification

Dimensions considered:

- dimension data: data used for training, e.g.,
  - data collection
  - data validation
  - feature extraction
  - poisoning removal
  - ...
- dimension process: training process
- dimension model: ML model in operation

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Multi-Dimensional ML Certification

Adaptation of our multi-dimensional certification scheme for ML certification

Dimensions considered:

- dimension data: data used for training
- dimension process: training process, e.g.,
  - training process implementation
  - boosting
  - strengthening
  - ...
- dimension model: ML model in operation

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Multi-Dimensional ML Certification

Adaptation of our multi-dimensional certification scheme for ML certification

Dimensions considered:

- dimension data: data used for training
- dimension process: training process, e.g.,
  - training process implementation
  - boosting
  - strengthening
  - ...
- dimension model: ML model in operation

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Multi-Dimensional ML Certification

Adaptation of our multi-dimensional certification scheme for ML certification

Dimensions considered:

- dimension data: data used for training
- dimension process: training process
- dimension model: ML model in operation, e.g.,
  - prediction inspection
  - memory and CPU/GPU usage
  - ...

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Multi-Dimensional ML Certification

Adaptation of our multi-dimensional certification scheme for ML certification

Dimensions considered:

- dimension data: data used for training
- dimension process: training process
- dimension model: ML model in operation, e.g.,
  - prediction inspection
  - memory and CPU/GPU usage
  - ...

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Multi-Dimensional ML Certification: a Bit of Formalization

Certification model $\mathcal{CM}=\langle p, ToC, \mathcal{E}, \mathcal{F}\rangle$

Each dimension induces a view:

- $\mathcal{V}_d=\langle p_{D_d}, ToC_{D_d}, \mathcal{E}_{D_d}, \mathcal{F}_{D_d}\rangle$
- $\mathcal{V}_p=\langle p_{D_p}, ToC_{D_p}, \mathcal{E}_{D_p}, \mathcal{F}_{D_p}\rangle$
- $\mathcal{V}_m=\langle p_{D_m}, ToC_{D_m}, \mathcal{E}_{D_m}, \mathcal{F}_{D_m}\rangle$

M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

ML Certification:
Reference Example on Property Robustness

Background

Poisoning attacks: attacks carried out at training time with the aim of

- degrade the performance of the ML model (*untargeted poisoning*)
- misclassify specific data points at inference time (*targeted poisoning*)
- by altering the training set
  - label flipping: change the labels

Background

Poisoning attacks: attacks carried out at training time

ML models can be protected by

- removing or sanitizing poisoned data points (dataset-level defense)
- strengthening the ML model (model-level defense)
- ...

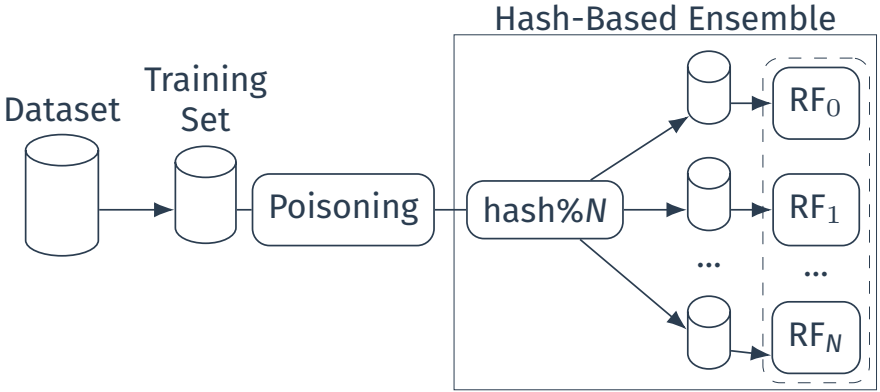
Our Approach

Our approach: use an ensemble of ML models instead of an individual ML model (model-level defense)

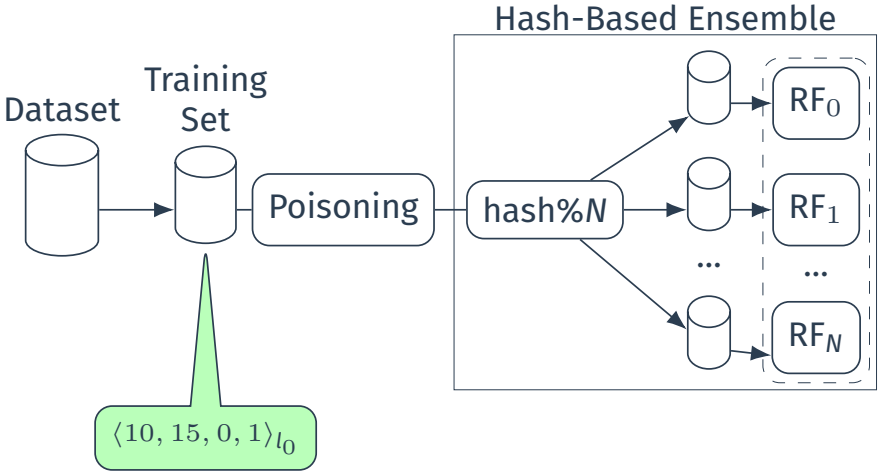
1. replace the individual ML model with an ensemble of models
2. split the dataset into partitions according to some strategies
3. train independently each ML model of the ensemble with a partition
4. retrieve the final prediction with majority voting

M. Anisetti, C. A. Ardagna, A. Balestrucci, N. Bena, E. Damiani, and C. Y. Yeun. "On the Robustness of Random Forest Against Untargeted Data Poisoning: An Ensemble-Based Approach". In: *IEEE Transactions on Sustainable Computing* (2023)

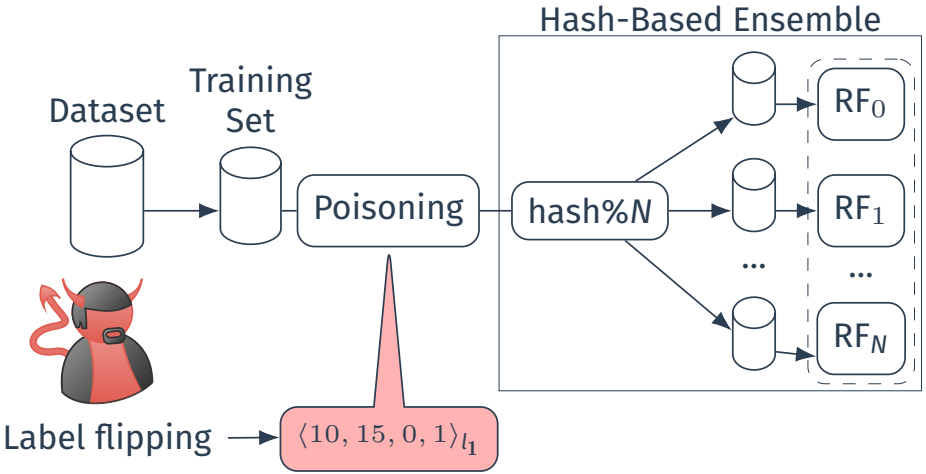
Our Approach: Robust Training



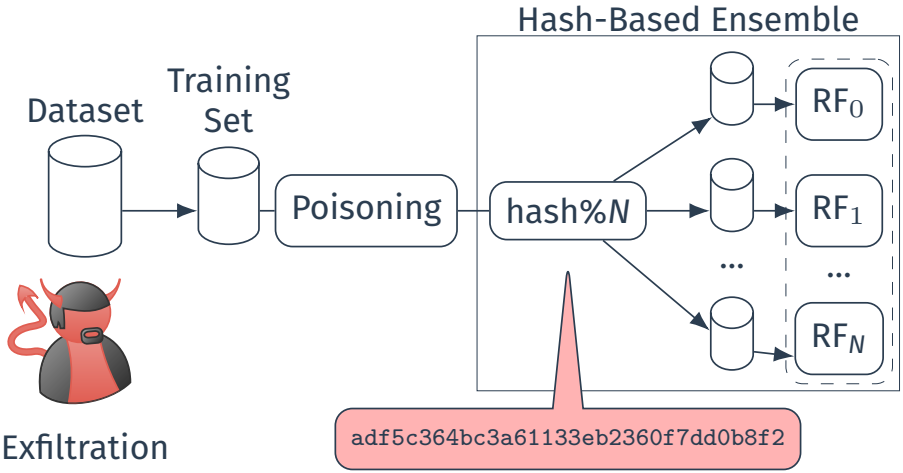
Our Approach: Robust Training



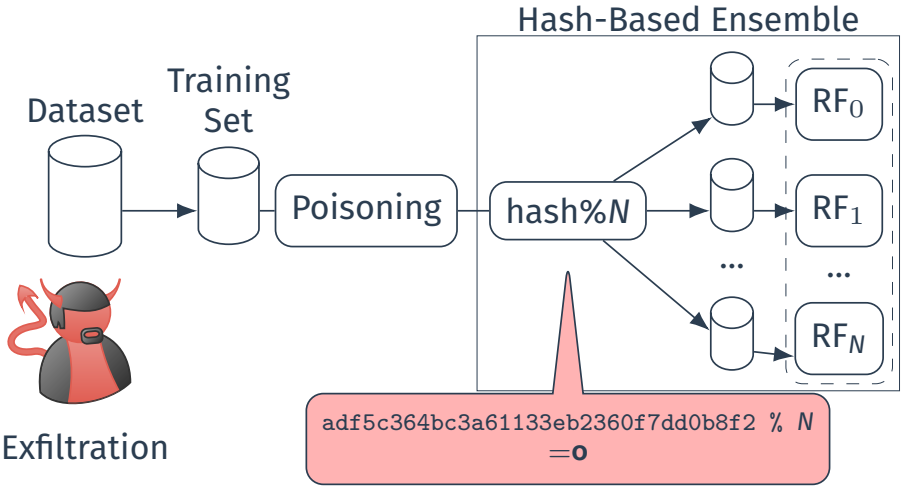
Our Approach: Robust Training



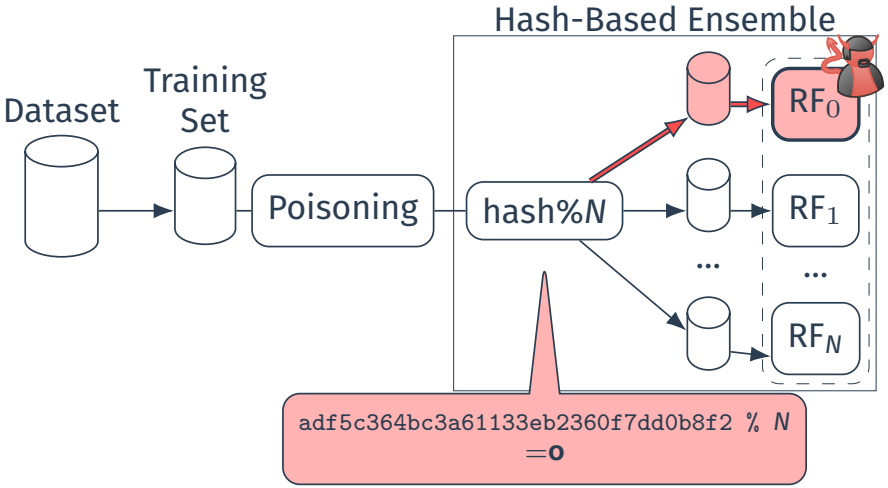
Our Approach: Robust Training



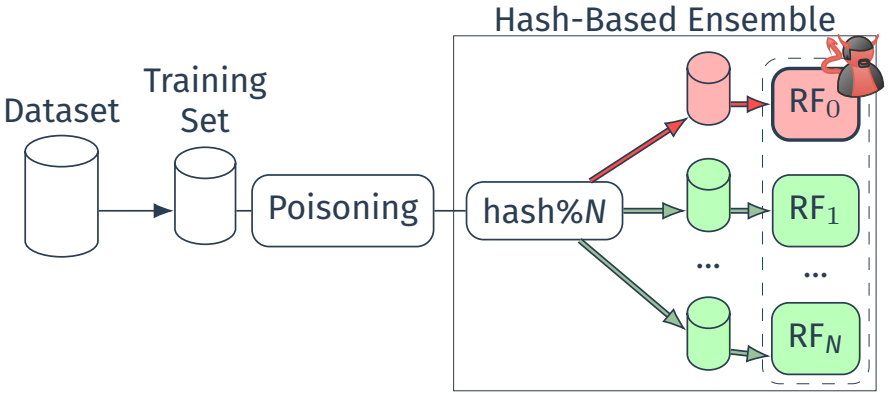
Our Approach: Robust Training



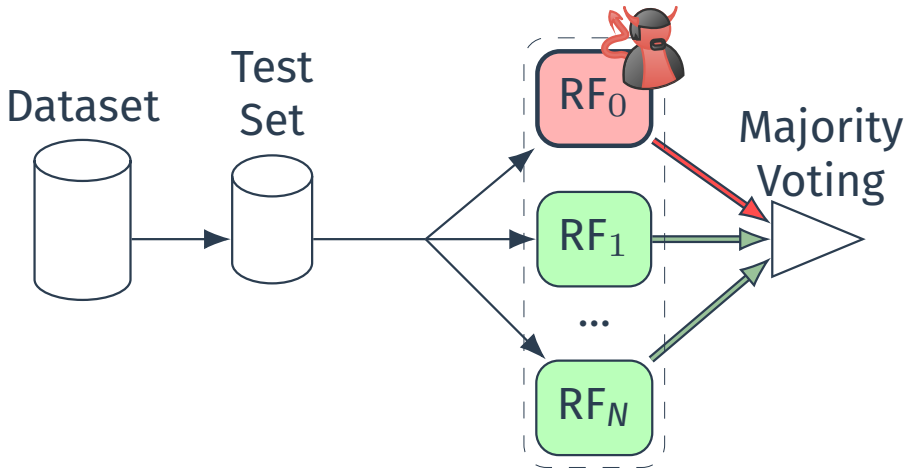
Our Approach: Robust Training



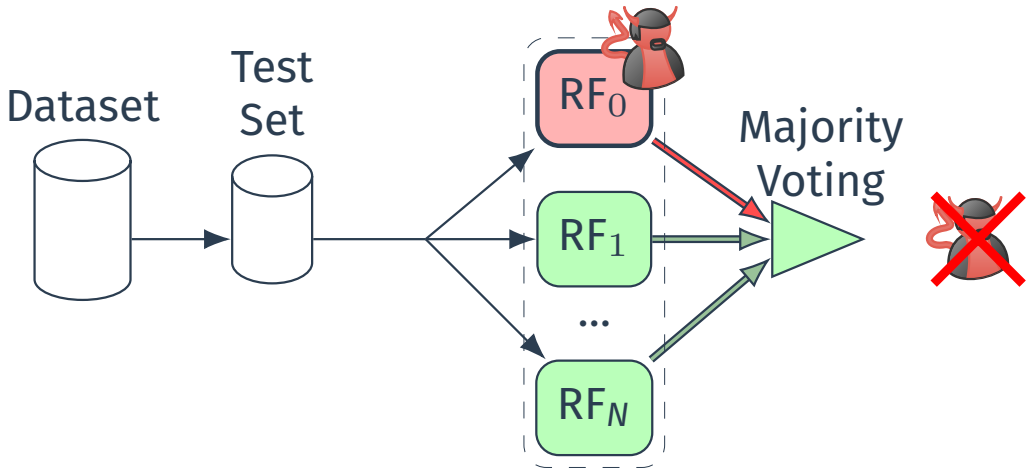
Our Approach: Robust Training



Our Approach: Testing and Inference



Our Approach: Testing and Inference



Does it Work?

We train the plain and ensemble models on the original and poisoned datasets

Name	N° data points (N° per class)	N° features	Sparsity (%)	N° data points (Preproc.)	Training set size (N° per class)	Test set size (N° per class)
Musk2 (M2)	6,598 (5,581/1,017)	166	0.28	2,034	1,628 (818/810)	406 (199/207)
Spambase (SB)	4,061 (1,813/2,788)	57	77.44	3,626	2,901 (1,443/1,458)	725 (370/355)
Diabetic Retinopathy Debrecen (DR)	1,151 (540/611)	19	10.41	1,080	864 (433/431)	216 (107/109)

Does it Work?

We train the plain and ensemble models on the original and poisoned datasets

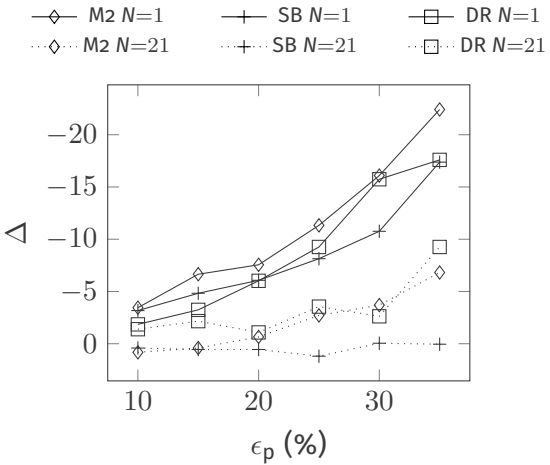
Dataset	Random Forest	Ensemble
M2	-11.248	-3.154
SB	-8.3912	-0.4429
DR	-8.9508	-3.0685
AVG	**-9.53**	**-2.2218**

Does it Work?

We train the plain and ensemble models on the original and poisoned datasets

We measure

- ACC(D)
- ACC($\widetilde{D}$)
- Δ=ACC($\widetilde{D}$)−ACC(D)



Can We Certify It? (1)

Scenario: ML-based malware detector

Dataset and inference:

- dynamic malware detection based on observed behavior
- each feature is sequence of three consecutive system calls
- the value of the feature is the number of times the system call has been invoked

Can We Certify It? (1)

Scenario: ML-based malware detector

Dataset:

Name	N° data points (N° per class)	N° features	Sparsity (%)	N° data points (Preproc.)	Training set size (N° per class)	Test set size (N° per class)
Android Malware (AM)	18,733 (11,479/7,254)	1,000	92.37	14,508	11,607 (5,776/5,831)	2,901 (1,478/1,423)

Can We Certify It? (1)

Scenario: ML-based malware detector

Goal:

- certify different versions of the applications
- varying the number of random forest models N
- $\Longrightarrow$ need to prepare one $\mathcal{CM}$ for each version
- property: *robustness against untargeted poisoning attacks*

Can We Certify It? (2)

We prepare 10 $\mathcal{CM}$, varying the number of random forest models N in $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$

Dimension process induces a view over the $\mathcal{CM}$ as $\mathcal{V}_p = \langle p_{D_p}, ToC_{D_p}, \mathcal{E}_{D_p}, \mathcal{F}_{D_p} \rangle$

- p_{D_p} is defined as the *correct usage of the strengthening technique*
  - the ensemble has a given number of random forest models N
  - the hash function is MD5 or SHA1
- ToC_{D_p} is the training process with a given value of N and hash function MD5
- $\mathcal{E}_{D_p}$ inspects the code and configurations of the training process
- $\mathcal{F}_{D_p}$ requires the all collected evidence is successful

Can We Certify It? (2)

We prepare 10 $\mathcal{CM}$, varying the number of random forest models N in $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$

Dimension model induces a view over the $\mathcal{CM}$ as $\mathcal{V}_m = \langle p_{D_m}, ToC_{D_m}, \mathcal{E}_{D_m}, \mathcal{F}_{D_m} \rangle$

- p_{D_m} is defined as *the loss in accuracy between the ML model trained on poisoned and original datasets is bounded by a given threshold* t^{D_m}

$$\min(\Delta, 0) \geq t^{D_m}$$

with $t^{D_m} = -2.5$

- ToC_{D_m} is the trained ensemble of random forest models with a given number of random forest models N and a specific hash function

Can We Certify It? (2)

We prepare 10 $\mathcal{CM}$, varying the number of random forest models N in $\{3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$

Dimension model induces a view over the $\mathcal{CM}$ as $\mathcal{V}_m = \langle p_{D_m}, ToC_{D_m}, \mathcal{E}_{D_m}, \mathcal{F}_{D_m} \rangle$

- $\mathcal{E}_{D_m}$ implements the evaluation process
  - executes the process 5 times on each percentage of poisoned points $\epsilon_p \in [5, 35]$ step 5
  - retrieves the average over the executions and different percentages, and compares it with t^{D_m}
- $\mathcal{F}_{D_m}$ requires the all collected evidence is successful
  - the averaged value of Δ is bounded by t^{D_m}

Can We Certify It? (3)

Results

- dimension data: successful in all views
- dimension model: successful in views (i.e., versions of the applications) with $N > 5$

ϵ_p	Number of random forest models N in the ensemble									
	3	**5**	**7**	**9**	**11**	**13**	**15**	**17**	**19**	**21**
AVG	-4.869	-2.725	-1.785	-1.303	-0.967	-1.072	-0.577	-0.616	-0.5	-0.239
	93.673	95.724	96.262	96.768	96.919	96.756	97.103	96.867	96.811	96.865

- $\mathcal{F}$ requires evidence to be successful in all views
- $\Longrightarrow$ we can release a certificate for the application versions with $N > 5$

Challenges, Ongoing, and Future Work

Our Manifesto for Research on Certification

Research direction	Challenge	Timeline
(1): on-functional property definition	C1.1: Property definition	M
	C2.1: Multi-layer service composition	S, M
	C4.2: Certification-based system life cycle	M, L
(2): Behavior-based certification	C1.2: Target modeling	M
	C2.1: Multi-layer service composition	S, M
	C2.3: Dishonest behavior	M
	C4.1: Increase automation	S, M
	C4.2: Certification-based system life cycle	M, L
(3): Trustworthy evidence management	C2.2: Evidence lineage	M
	C4.3: Reduce reliance on blind trust	M

Research direction	Challenge	Timeline
(4): Certification for ML	C1.1: Property definition	M
	C1.2: Target modeling	M
	C2.1: Multi-layer service composition	S, M
	C3.1: Property and target definition	M, L
	C3.2: Certification process modeling	M, L
	C3.3: ML pipelines	L
(5): ML-based automation	C1.1: Property definition	M
	C1.2: Target modeling	M
	C2.3: Dishonest behavior	M
	C4.1: Increase automation	S, M
(6): *DevCertOps* and beyond	C1.3: Integration of development and certification processes	M, L
	C2.1: Multi-layer service composition	S, M
	C4.1: Increase automation	S, M
	C4.2: Certification-based system life cycle	M, L

Research Directions

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Research directions guiding the definitions of novel certification techniques for modern distributed systems

Research Direction: Behavior-Based Non-Functional Property

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Challenges

- Existing non-functional properties
- Certification evaluation still relies on precise and human-made system modeling
  - but system boundaries are dynamic (lack of automation)
- Evidence management and collection still rely on static processes

Research Direction: Behavior-Based Non-Functional Property

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

- Flexible definition of properties based on system behavior
- Model expected system behavior and compare the retrieved behavior against it in a continuous fashion and adapting to system changes
- Trustworthy, human-readable evidence management and collection

M. Anisetti, C. A. Ardagna, and N. Bena. "Multi-Dimensional Certification of Modern Distributed Systems". In: *IEEE Transactions on Services Computing* 16.3 (2023); M. Anisetti, C. A. Ardagna, and N. Bena. "Continuous Certification of Non-Functional Properties Across System Changes". In: *Proc. of ICSOC 2023*. Rome, Italy, 2023

Research Direction: Certification of ML

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Challenges

Certification schemes are designed for deterministic systems that can be inspected or tested

- cannot model and certify a ML-based service whose behavior is unpredictable
- cannot be limited to run-time model evaluation

Research Direction: Certification of ML

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Novel building blocks for the certification of ML-based systems

- novel definition of non-functional property
- evaluation based on observed predictions or explainability
- along the complete ML pipeline and towards the complete ML-based system

M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero. "A Methodology for Non-Functional Property Evaluation of Machine Learning Models". In: *Proc. of MEDES 2020*. Abu Dhabi, UAE, Nov. 2020; M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. "Rethinking Certification for Trustworthy Machine Learning-Based Applications". In: *IEEE Internet Computing* (2023)

Research Direction: ML-Based Automation

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Challenges

Certification still relies on error-prone and expensive manual activities

- lack of automation
- reliance on precise and human-made system modeling
  - but system boundaries are dynamic

Research Direction: ML-Based Automation

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Use ML to boost the automation of certification activities

- automatically infer target system's behavior and properties
- automatically derive the corresponding evaluation

Research Direction: DevCertOps and Beyond

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Challenges

Certification is still seen as one-time, post-deployment activity

- lack of tight integration within the system life cycle
- lack of *usage* of certificates after their issuing

Research Direction: DevCertOps and Beyond

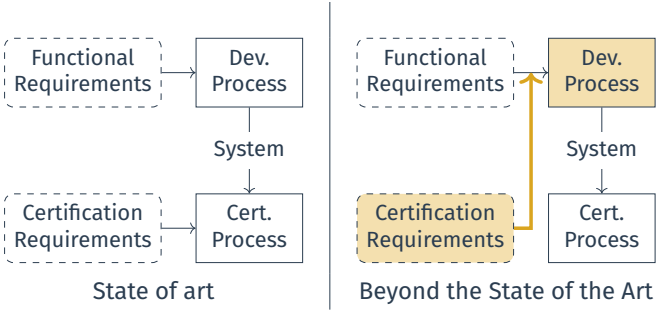
- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond

Integration of system development life cycle and certification life cycle

- certify all development/deployment artifacts
  - *shift certification to the left*
- certification part of the process driving system evolution

Research Direction: DevCertOps and Beyond

- (1): non-functional property definition
- (2): behavior-based certification
- (3): trustworthy evidence management
- (4): certification of ML
- (5): ML-based automation
- (6): *DevCertOps* and beyond



C. A. Ardagna, N. Bena, and R. M. de Pozuelo. "Bridging the Gap Between Certification and Software Development". In: *Proc. of ARES 2022*. Vienna, Austria, Aug. 2022

Conclusions

Certification techniques evolved following the IT evolution

They face new challenges to meet the peculiarities of modern distributed systems

- multi-dimensional certification to retrieve accurate certificates
- continuous certification to ensure that certificates remain valid
- ML certification to certify ML-based applications

References I

[1] M. Anisetti, C. Ardagna, N. Bena, and E. Damiani. “Stay Thrifty, Stay Secure: A VPN-based Assurance Framework for Hybrid Systems”. In: *Proc. of SECRYPT 2020*. Paris, France, July 2020.

[2] M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero. “A Methodology for Non-Functional Property Evaluation of Machine Learning Models”. In: *Proc. of MEDES 2020*. Abu Dhabi, UAE, Nov. 2020.

[3] M. Anisetti, C. A. Ardagna, A. Balestrucci, N. Bena, E. Damiani, and C. Y. Yeun. “On the Robustness of Random Forest Against Untargeted Data Poisoning: An Ensemble-Based Approach”. In: *IEEE Transactions on Sustainable Computing* (2023).

[4] M. Anisetti, C. A. Ardagna, and N. Bena. “Continuous Certification of Non-Functional Properties Across System Changes”. In: *Proc. of ICSOC 2023*. Rome, Italy, 2023.

[5] M. Anisetti, C. A. Ardagna, and N. Bena. “Multi-Dimensional Certification of Modern Distributed Systems”. In: *IEEE Transactions on Services Computing* 16.3 (2023).

[6] M. Anisetti, C. A. Ardagna, N. Bena, and R. Bondaruc. “Towards an Assurance Framework for Edge and IoT Systems”. In: *Proc. of IEEE EDGE 2021*. Guangzhou, China, Dec. 2021.

References II

[7] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. “An Assurance Framework and Process for Hybrid Systems”. In: *Proc. of ICETE 2020 - Revised and Selected Papers*. Paris, France, Nov. 2020.

[8] M. Anisetti, C. A. Ardagna, N. Bena, and E. Damiani. “Rethinking Certification for Trustworthy Machine Learning-Based Applications”. In: *IEEE Internet Computing* (2023).

[9] M. Anisetti, C. A. Ardagna, N. Bena, and A. Foppiani. “An Assurance-Based Risk Management Framework for Distributed Systems”. In: *Proc. of IEEE ICWS 2021*. Chicago, IL, USA, Sept. 2021.

[10] M. Anisetti, C. A. Ardagna, N. Bena, V. Giandomenico, and G. Gianini. “Lightweight Behavior-Based Malware Detection”. In: *Proc. of MEDES 2023*. Heraklion, Greece, May 2023.

[11] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi. “A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification”. In: *IEEE TSC* 13.1 (2020).

[12] M. Anisetti, N. Bena, F. Berto, and G. Jeon. “A DevSecOps-based Assurance Process for Big Data Analytics”. In: *Proc. of IEEE ICWS 2022*. Barcelona, Spain, July 2022.

[13] C. A. Ardagna, R. Asal, E. Damiani, and Q. Vu. “From Security to Assurance in the Cloud: A Survey”. In: *ACM Computing Surveys* 48.1 (2015).

References III

[14] C. A. Ardagna and N. Bena. “Non-Functional Certification of Modern Distributed Systems: A Research Manifesto”. In: *Proc. of IEEE SSE 2023*. Chicago, IL, USA, July 2023.

[15] C. A. Ardagna, N. Bena, C. Hebert, M. Krotsiani, C. Kloukinas, and G. Spanoudakis. “Big Data Assurance: An Approach Based on Service-Level Agreements”. In: *Big Data* 11.3 (2023).

[16] C. A. Ardagna, N. Bena, and R. M. de Pozuelo. “Bridging the Gap Between Certification and Software Development”. In: *Proc. of ARES 2022*. Vienna, Austria, Aug. 2022.

[17] N. Bena, R. Bondaruc, and A. Polimeno. “Security Assurance in Modern IoT Systems”. In: *Proc. of IEEE VTC 2022-Spring*. Helsinki, Finland, June 2022.

[18] E. Damiani and C. A. Ardagna. “Certified Machine-Learning Models”. In: *Proc. of SOFSEM 2020*. Limassol, Cyprus, Jan. 2020.