Report on Cybersecurity challenges and opportunities in critical systems

The 87th Meeting of the IFIP WG 10.4 on Dependable Computing and Fault-Tolerance

Praia do forte

10/02/2025

3 presentations

- Session chair: Pascal Felber
- Confidential Encoded Processing/Computing to Build Safety Critical Systems
  - Christof Fetzer
- Research Challenges at the Intersection of Cybersecurity and Safety
  - Bruno Crispo
- Dependable software engineering: can we increase trust in our components?
  - Marcelo Pasin

Confidential Encoded Processing/Computing to Build Safety Critical Systems

- Confidential Computing key principles and main benefits
- Problem
  - Large number of vulnerabilities and corresponding fixes that need to be installed
  - Migrating an enclave VM is difficult due to (among others) keys associated with hosts
- Presented Solution:
  - A mechanism for updating/moving hosts, VMs, Pods
  - Secret provisioning in Scone

Q&A:

- Q: Adoption of Confidential Computing: How did you convince customers to invesrt in CC
- A: German health regulation has put CC as a requirement

Research Challenges at the Intersection of Cybersecurity and Safety

Context
- Amount of attacks is increasing in mission critical systems
- There are many security off the shelf solutions

Problem
- Their assumptions are not well explicited (example of TLS and traffic analysis)
- On the importance of choosing the most appropriate (off the shelf) security solution (example of stardard ISO15118)

TEEs as a good off the shelf solution to enforce confientiality and integrity but TEE solutions are very heterogeneous and lack interoperability
Intrusion detection (based on classifcation, anomaly detection etc) is a good alternative

Problem: what do we do in case of intrusion detection
- Automated response is not satisfactory
- Remote healing of a compromised device is very challenging

Long Q&A session about the interaction between security and safety

Dependable software engineering: can we increase trust in our components?

Problem

- How much we can trust current software distributions

Problem quantification

- On a 1000 analyzed dockerhub repositories that have about 400 dependencies on average each, half of the projects have unpatched dependencirs that have known CVEs

Solution key principles

- Add Treacability and attest the code execution (using TEEs)
- A traceable software supply chain with annotations -> use TEEs to make sure the annotations are generated correctly

Q&A

How much do you need to trust the compiler?

Is transparency a good thing?