Hypothetical Trust and Attack Models

Mariano Ceccato (1), Christian Collberg (2), Paolo Tonella (1)

(1) ITC-irst, Trento, Italy

(2) University of Arizona, USA

ceccato@itc.it, tonella@itc.it, collberg@cs.arizona.edu

Outline

- Architecture of the software-only solution.
- Sources of trust.
- Possible attacks.
- Open issues.

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

The remote entrusting problem

Remote software authentication: ensuring a trusted machine (server) that an untrusted host (client) is running a “healthy” version of a program *P*:

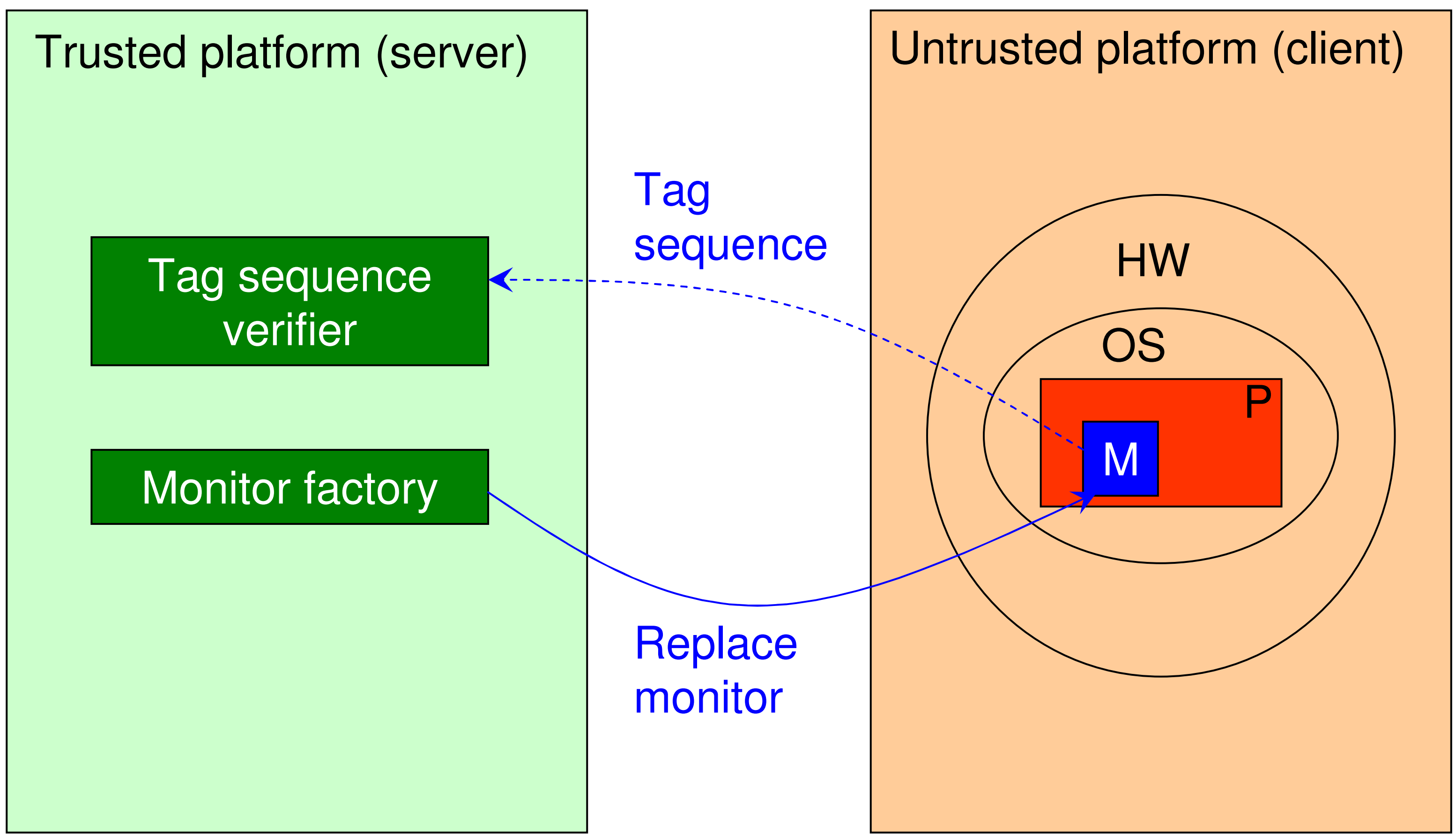
- The program is unadulterated.
- It is executed on top of unadulterated HW/SW.
- The execution process is not manipulated externally.

The distinctive feature of *remote* entrusting is that the authenticated software needs to communicate over the network with the trusted machine to work properly.

Entrusting remote applications

- An increasing number of applications depend on services provided over the network.
- Often service providers need to “entrust” their clients and assume they do not act maliciously.
- This is critical for applications involving:
  - Trading (eCommerce), bidding, gaming.
  - eGovernment, eVoting.
  - Distributed (Grid) computing.
  - Protocol implementations.
  - Content and data protection.

Reference architecture



CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Sources of trust

Authenticity verification

The monitor *M* should verify:

- Text and data segments of *P* as loaded in memory.
- Libraries used by *P*.
- The execution environment (HW, OS, execution process, etc.).
- Results of specific computations or assertions.

Tag sequence generation

The monitor *M* sends the server an authenticity tag sequence as evidence of healthy execution:

- Tags have limited time validity.
- A secret key, hidden into M itself, is used to generate them.
- If no tag or an incorrect tag is received by the server, the client is considered untrusted and no service is delivered to it.

Replacement

To give attackers a limited time to succeed, the monitor *M* is periodically replaced:

- The duration depends on the estimated reverse engineering complexity, assuming humans are necessarily involved in the process.
- The monitor factory should generate highly independent monitors.

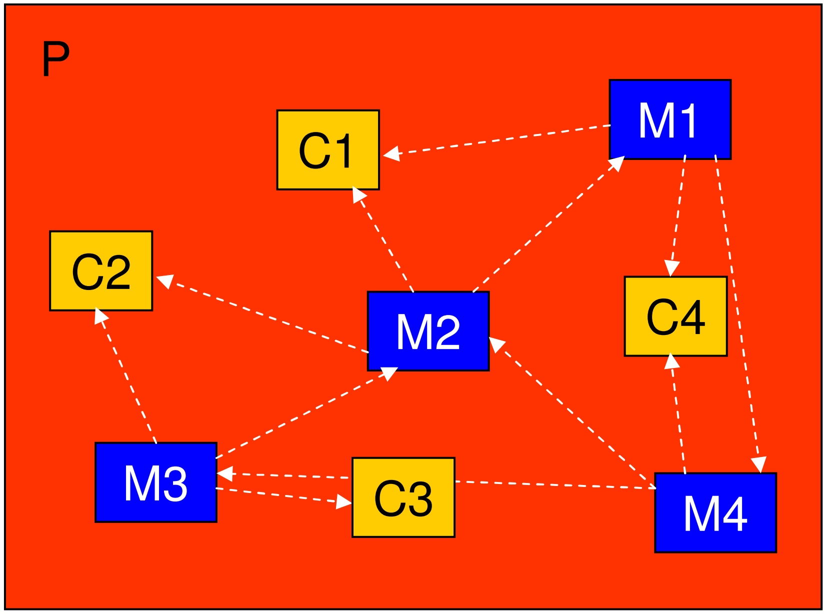
Code obfuscation

To increase the resistance to reverse engineering, the code is obfuscated:

- Opaque predicates based on conditions that are hard to analyze statically (e.g., involving pointer structures) could be used.

Additional sources of trust

- **Self checking monitor:** *M* checks itself before checking *P*.
- **Network of trust:**



- **Tags include data verified by server:** authenticity verification is no longer local to *M*
- **Server sends challenge *C* to client:** tag generation and authenticity verification depend on *C*.

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Attacks

Assumptions on attackers

A malicious user can:

- Install any software on the client.
- Read and write memory locations, processor registers and files.
- Observe and modify the network traffic.
- Modify *P* and *M*, both on disk and in memory.
- Use any available code analysis tool.
- Take advantage of tracers, emulators and debuggers.
- Tamper with libraries, operating system and hardware.

A malicious user cannot:

- Access and tamper with the trusted server.
- Know the software/hardware configuration of the server.

Classes of attacks

1. Reverse engineering attack.
2. Execution environment attack.
3. Cloning attack.
4. Differential analysis attack

Reverse engineering attacks

Important functionalities and data structures are located and altered maliciously in *P* and *M*:

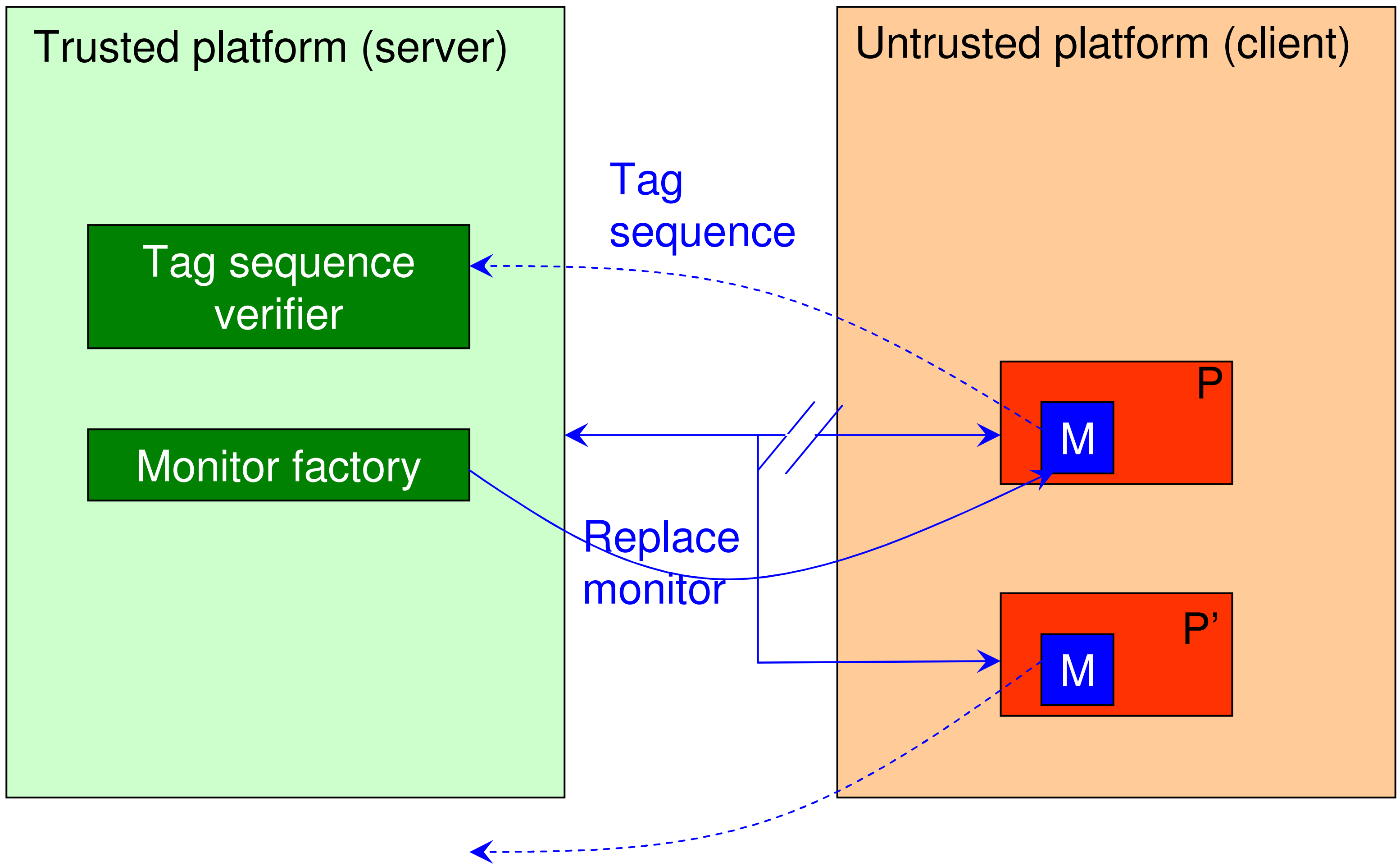
- Tag sequence generator.
- Authenticity checking functions.
- Secret keys.
- Input data (e.g., passed to checking functions).
- Output data (e.g., returned by checking functions).

Execution environment attacks

P is run on an emulator, in debug mode or is interpreted by an adulterated virtual machine:

- Memory locations, call stack, program counter and parameters can be altered dynamically.
- Dynamic libraries can be altered maliciously.
- Input and output values can be replaced on-the-fly.

Cloning attack

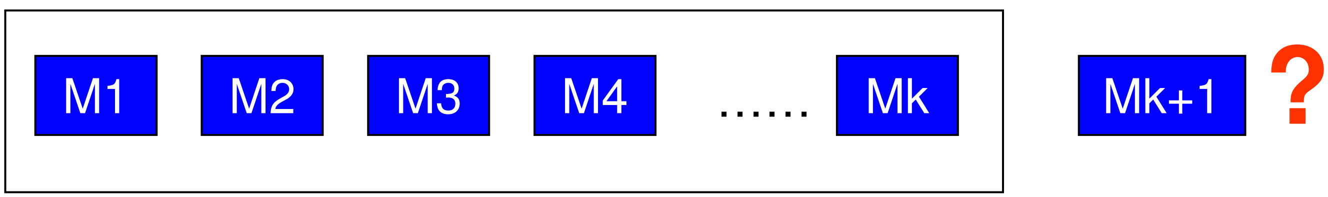


This attack is ineffective if tag sequence includes computation data.

Differential analysis attack

The attacker gathers information about *M* by comparing the sequence of monitors delivered by the monitor factory in the past:

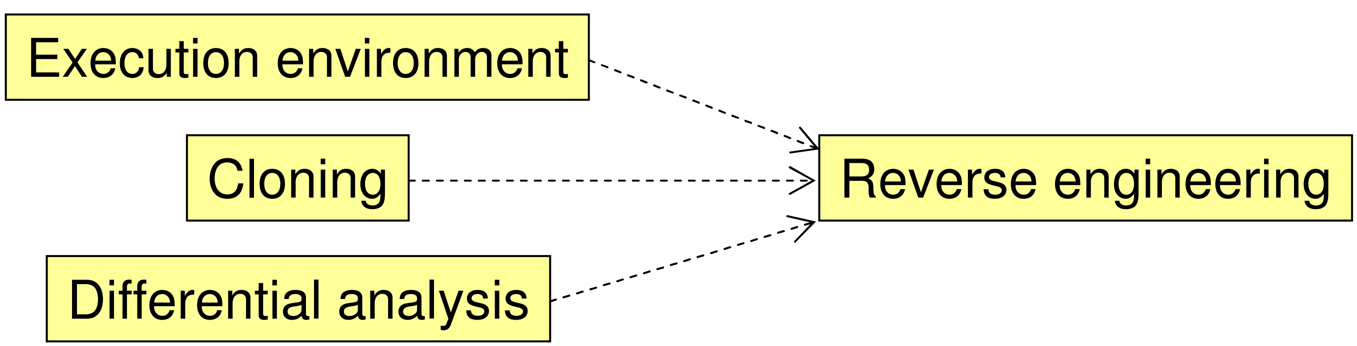
- If the strategy used by the monitor factory is even only partially understood, the time necessary to break new monitors might be reduced, eventually allowing the attacker to break a yet valid monitor.



Dependencies among attacks

Execution environment, cloning and differential analysis attacks are effective only if combined with reverse engineering, which provides information on:

- Where functionalities and data structures are implemented.
- When to intercept the execution.
- How to alter it.



Analysis of attack resistance

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12
T1	X											
T2												
T3									X			
T4										X		
T5												
T6	X											
T7												
T8												
T9	X											X
T10												
T11												
T12												

T8: Monitor replacement

T1: M checks P text and data segment

T2: M self checks itself before checking P

T3: M checks libraries used by P

T4: M checks execution environment

T5: M checks OS and HW

T6: M checks results of computation

T7: Secret key used to generate tag sequence
A3: Replace tag sequence generator

A5: Modify input before call on env.
A6: Modify output before return M/P
A7: Modify output before return on env.
A8, A9: Replace HW/OS; replace dynamic libraries
A10: Tampered execution (debug mode)

Open issues (1)

- Protection against reverse engineering:
  - Code obfuscation metrics
  - Beyond code obfuscation?
- Monitor factory:
  - How to construct independent monitors automatically?
  - What is the minimum set of functionalities of a monitor?
- Network of trust:
  - What replacement frequency can we achieve with it?
- Tags include output data:
  - How to avoid/minimize replication of the computation on the server?

CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

Open issues (2)

- Server sends challenge to client:
  - How does it complement/substitute monitor replacement?
- Tag sequence generator:
  - How to hide secret key into the monitor's code?
- Execution environment:
  - How to check sanity of HW and execution mode?
  - How to recognize parallel execution of clone process?

ITC irst
CENTRO PER LA RICERCA SCIENTIFICA E TECNOLOGICA

Conclusions

- The reference architecture alone is very fragile and should be augmented with:
  - Network of trust
  - Challenge from server
  - Data embedded into tag sequence
- Replacement and monitor factory are at the core of remote entrusting
  - Independent monitor generation is crucial

Proposal for a new reference architecture