WP4 – Trust and Security Analysis
Y3 report

Cataldo Basile
<cataldo.basile@polito.it>
WP4 goal and tasks

- Task T4.1 – Trust analysis of SW-based method (M6-M36)
- Task T4.2 – Trust analysis of HW/SW-based method (M12-M36)
- Task T4.3 – Reverse engineering complexity (M0-M36)
- Task T4.4 – Comparative analysis with TC from OS perspective (M12 – M36)
- Task T4.5 – Remote entrusting and Internet secure protocols (M6-M30)
Trust analysis of SW-based and HW/SW-based solutions

- security analysis
  - the same analysis for SW-based and HW/SW-based techniques
    - the trusted HW behaves exactly as the trusted entity behaves (trusted black box assumptions)
  - behavioral attacker model, automatic vs. manual attacks
  - model of manual attacks: micro-cycle of changes
    - man in the loop
  - security analysis of RE-TRUST techniques and solutions

- attacking programs in remote entrusting scenario is harder than maintaining source code
  - skype attack needed two years

- experimental test plan
  - task shared between T4.1&T4.2 and T4.3
Security analysis: attacker behaviour

- the attacker will try
  - first automatic attacks $\rightarrow$ require less effort
    - using automatic attack & reverse engineering tools
  - then manual attacks
    - the attacker must understand the code of P
  - we assume no trivial protections
RE-TRUST solutions: security objectives

- develop solutions for which automatic tools are not effective
- if automatic attacks cannot be avoided,
  - computational resources of the attacker must not be enough to perform a successful attack during the available time
    - focus on increasing the computational complexity
- otherwise, the time needed to mount a manual attack must be bigger than the time available to the attacker
  - human in the loop
  - increase the complexity of program’s code understanding
  - improve the detective capability of trusted entity
  - standard analysis methodologies fail
    - no quantitative information or pseudo-quantitative or qualitative (from empirical studies)
Attacker behaviour: revised micro-cycle

- Attack goal

- Automatic attack tools

- Reverse engineering tools

- Complex attacks = micro-cycle

- Reverse engineering tools

- Successful attack

- Detected attack

- Interaction with T

- Off-line testing

- Impact analysis

- Local impact analysis

- Anticipating T detective techniques

- Attack implementation

- MANUAL ATTACK
Reverse engineering complexity

- evaluation of tools that may serve for manual attacks
- de-compilers
  - de-compilers unable to reverse engineer the entire source code
    - data and often indistinguishable from instructions
  - optimizing compilers
  - even worse for multi-core/multi-processor architectures
- de-compile $\rightarrow$ modify source $\rightarrow$ compile cycles
  - impossible to perform, usually direct binary modification
- reverse engineer tools loose many useful information
  - variable names (speed vs. a0001)
- reverse engineering tools cannot provide a completely automatic support
Experimental analysis

- approach from experimental software engineering

To define a set of experiments needed to evaluate the effect of RE-TRUST techniques to the protection of programs in remote entrusting scenario.

**Main factors**
- Define a factors/parameter that characterize the solution:
  - Choose one
  - Fix the others (define a representative set)
  - Repeat for each parameter

**Hypothesis II**
- Is the parameter X of RE-TRUST solution X effective in protecting software in remote entrusting paradigm?

**Co-factors**
- Define environmental parameters:
  - Programs
  - Humans

**Experiment-specific co-factors**

---

T4.1
T4.2
T4.3

**Experiment results**
Software Obfuscation

- theoretical impossibility to of general a purpose obfuscator [Barak2001]
- widely used to prevent / limit malicious reverse engineering
- transforming a program into an equivalent one
  - maintaining its semantics
  - harder to reverse engineer

How much harder? Never investigated in literature
Results and future work

- different attack efficiency on clear or obfuscated code
  - obfuscated code is up to 4 times more complex to understand
- reduce the gap between experienced programmers and beginners
  - the gap is very high for clear code
- next experiment: comparison of different obfuscation techniques
  - experiment prepared, result just collected, not yet analysed
  - two already published papers and one paper will come...
**Comparative analysis of TC from OS perspective**

- focus on Trusted Computing approach
  - TC solutions need extensive platform changes (HW/SW)
  - commercial adaption and deployment of TC technology limited
  - remote attestation by TCG platform has some shortcomings
  - difficult to manage large number of possible configurations
  - only load-time assurance about software integrity
  - application level attestation with TC requires secure operating system or hypervisor

- RE-TRUST gives runtime assurance about software integrity

- possible synergies
  - use TCG attestation to verify hypervisor and virtual machine image
  - hypervisor uses RE-TRUST technique to monitor runtime integrity of application
OS issues

- assume RE-TRUST as perfect solutions
  - programs that do not use external code are protected by RE-TRUST
- the program needs external code
  - libraries, HW drivers, etc.
- RE-TRUST solutions may protect software even when the execution environment is compromised
  - program dependent
- environmental attacks
  - replacement → partially effective (if links to libraries are also changed)
  - code splitting → not effective
  - obfuscation → partially effective (difficult to understand what to change)
  - local/remote integrity code verification → not effective even if attacks are not trivial
  - use state-based remote attestation
    - to check variables whose values depend from external code
OS issues

- assume RE-TRUST as perfect solutions
  - programs that do not use external code are protected by RE-TRUST

- the program needs external code
  - libraries, HW drivers, etc.

- RE-TRUST solutions may protect software even when the execution environment is compromised
  - program dependent

- environmental attacks
  - replacement $\rightarrow$ partially effective (if links to libraries are also changed)
  - code splitting $\rightarrow$ not effective
  - obfuscation $\rightarrow$ partially effective (difficult to understand what to change)
  - local/remote integrity code verification $\rightarrow$ not effective even if attacks are not trivial

  - use state-based remote attestation
    - to check variables whose values depend from external code
Analysis of interaction of RE-TRUST with security protocols

- Existing security protocols in Remote entrusting:
  - **TLS, SSL, DTLS**
    - mutual authentication and key establishment of the entrusting protocol
    - public keys management issues
  - **IPsec** not well-suited for entrusting protocol
    - host-to-host then application-to-application one (so it is more vulnerable via man-in-the-end-attack)
  - **Kerberos**
    - mutual authentication and key agreement
    - replicated servers multiple realms deployment
    - trust zone expansion problem
  - **Time-Stamp Protocol (TSP) and Network Time Protocol (NTP)**
    - to keep synchronized time on different hosts
  - **Public key management** becomes critical, if TLS or other solutions assume public keys is used
    - use trust models for public keys management
Analysis of interaction of RE-TRUST with security protocols

- remote entrusting (RE) for improving security protocols:
  - enhance the security protocols assuming long-running dynamically change different parameters
    - regularly update different aspects of encrypted interactions
      - algorithms, algorithm parameters, etc.
      - not only the keys
  - enhance Kerberos protocol to re-authenticate client and their software implementation
    - possibly using another key distribution centre

- remote attestation
  - enhance Kerberos ticket by adding RE approval of the client host
Conclusions

- RE-TRUST solutions are effective for protecting programs in remote entrusting paradigm
- The (unfair) power of the trusted entity is not fully understood
- Evaluation of complexity of reverse engineering is still an open question
  - Many factors and experts judgments support our positive impression
  - Dependency from other open issues (e.g., program’s complexity)
- RE-TRUST solutions interface with protocols and can join forces with the other trust initiatives (e.g., TCG)
- ...and many other research issues