Application Trustworthiness

ACSAC Panel

December 5, 2012

Rance DeLong

Application Trustworthiness

- What do I expect from applications?
  - Accomplish what I call on them to do, without wasting my time, trying my patience, or otherwise giving me nasty surprises.
  - I ***hope*** the developer is benevolent and competent.
  - I ***hope*** the developer protects me from nasty surprises.
  - I ***hope*** the application was not subverted by another party before (or after) I receive it.
  - I would like ***assurance*** instead of ***hope***.
- The developer may not be benevolent.
- The developer may not be competent.
- The platform may not be trustworthy.

Application Trustworthiness

Something that is trustworthy will always act in accordance with my expectations (e.g., a prior agreement), or according to my best interests.

Assume: Some application is trustworthy

X is trustworthy, i.e., 

is trustworthy

X must be endowed with the ability to act.

X is bounded and defined. Within the circle is X (all the parts of X), and outside the circle is Not X.

The actions of X are always in accordance with my expectations.

This presumes:

X is solely able to determine its actions (X is not coerced to act by Not X nor prevented from acting by Not X).

X is “aware” of the prior agreement, or able to “know” my best interests.

Application Trustworthiness

- No third party (i.e., Not X) may interfere with the actions of the application X if the application is solely to determine its own actions.
- An application is endowed with the ability to act by its developer.
- The developer may not be benevolent:

  The developer may intend the application to act contrary to my best interests.
- The developer may not be competent:

  The developer may not know, or be able to ensure, my best interests.
- Another party may have subsequently "brainwashed" or coerced (subverted) the application to act contrary to my best interests.

Application Trustworthiness

- However, it is noted that:
  - The operating system can interfere with the actions of any application.
  - The hardware can interfere with the actions of any application.
- The operating system and hardware are either X or they are Not X (viz., they are, or are not, part of an application).
  - By conventional understanding the operating system and hardware are not ***parts of*** any application.
- Therefore, no application can solely determine to act according to my expectations!
- Contradicts the premise “Some application is trustworthy.”
- We are forced either to consider the OS and the hardware to be part of the application, and to be certain they are trustworthy, or we my not consider any application to be trustworthy.

What to trust: App, OS, or Hardware

- The user tends to trust the whole lot.
- Can the trust burden be shifted among these parts?
- What if the OS and hardware ***are*** trustworthy?
- If we ***assume trustworthiness of OS and HW*** (they are correct and without subversion), what is necessary for an application to be trustworthy?
  - *The application must be correct and without subversion.*
- It still comes down to the application developer and the supply chain (for correctness and absence of subversion).
- At least in this case it is not ***logically impossible*** !

To trust the application

- Must trust that the result of the developer's efforts is a correct application.
- Must trust that the application is not subverted either before or while it is in use.
  - An application that contains vulnerabilities may be able to be subverted while it is in use.

Needed to trust an application

- Correct requirements
- Correct design
- Correct implementation
- Well-defined and safe implementation language
- Well-defined compiler and tools
- Configuration management of artifacts
- Integrity of delivery (supply chain)
- Integrity of installation, configuration and initialization
- Sounds a lot like an A1 or EAL 7 product!
- Surprise, surprise.

How can we trust an application?

- Current approach: “Secure coding”
  - Necessary because, without extreme care, *the implementation language may cause the application to provide interfaces enabling its own runtime subversion*!
- Near future: Increase attacker work factor
  - Software diversity to thwart buffer overflow and control-flow attacks, e.g., address space layout randomization, instruction set randomization, and compiler-provided diversity of application binaries
  - Continuous runtime policy monitoring and mediation
- Future: Better still
  - Trustworthy HW, OS, language, cf. DARPA CRASH
  - Fine-grained compartmentalization, cf. CRASH
  - Aspect-oriented automated policy weaving, cf. CRASH