



# SAFETY and SECURITY TWO SIDES of the SAME COIN?

---

**SAFECOMP Workshop**

**The Relationship between Safety and Security in  
Software-Based Systems**

**Robert B K Dewar**

**September 25th, 2008**

# Safety-Critical Software: Background

---

- **What is “safety critical” software?**
  - Failure can cause loss of human life or have other catastrophic consequences
- **How does safety criticality affect software development?**
  - Regulatory agencies require compliance with certification requirements
  - Safety-related standards may apply to the finished product, to the development process, or both
- **What exactly do we mean by failure?**
  - Informally: doesn't do what it is supposed to do
  - More precisely, implementation does not match specification
  - Note this assumes specification is complete and correct!



- **Level A**
  - Anomalous behavior  $\Rightarrow$  catastrophic failure condition
  - “prevent continued safe flight and landing”
- **Level B**
  - Anomalous behavior  $\Rightarrow$  hazardous / severe-major failure condition
  - “serious or potentially fatal injuries to a small number of ... occupants”
- **Level C**
  - Anomalous behavior  $\Rightarrow$  major failure condition
  - “discomfort to occupants, possibly including injuries”
- **Level D**
  - Anomalous behavior  $\Rightarrow$  minor failure condition
  - “some inconvenience to occupants”
- **Level E**
  - Anomalous behavior  $\Rightarrow$  no effect on aircraft operational capability or pilot workload

## More on Safety

---

- **Basically what we ask of a safety-critical program is that it operate completely correctly without any defective behavior.**
- **Of course that's what we would like to have be true of any program we write.**
- **In practice the basic approach for most SC software is:**
  - Use really careful procedures
  - Specify what you are doing really carefully
  - Test the heck out of the final code
- **DO-178B codifies these steps**
  - In particular it specifies exactly what is required for thorough testing, e.g. full coverage required, objectives all tested.

# Security-Critical Programs

- **A program which must be able to guarantee absolute security with respect to a stated set of critical properties.**
  - Even in the presence of (clearly characterized) intrusion attempts
- **Unlike Safety-Critical programs**
  - It is not required that Security-Critical programs work!
  - Just that they cannot be compromised with respect to well stated security objectives
  - For instance an OS that bombs and reboots in response to an intrusion attempt but fails to yield up critical data such as passwords may satisfy the security requirements.
  - A voting machine that crashes is OK, recording bad votes is not
- **Of course we would like them to work reliably**
  - But that's true of any program!

## More on Security-Critical Programs

---

- **The statement of security objectives is a critical part of the process**
  - This is different from the general issue of specification
  - Because it focuses only on required security properties
  - And goes beyond saying that the program implementation must simply follow the specifications for these properties
  - We must demonstrate resistance to intrusion or compromise from deliberate or accidental external effects.
- **Example of accidental compromise**
  - Entering a junk command should not bring down an OS
- **Example of deliberate intrusion**
  - Sending a series of packets to the OS on an ethernet line should not permit the discovery of any stored passwords

## An Example of A Security Property

- **An application running on an OS must not be able to crash the OS, by accidental or deliberate failure to follow required interface rules.**
- **A spectacular example of failure in this respect**
  - 1998, Smart Ship Project
  - Yorktown towed into port because of failure of all computers
  - Computer system using Windows NT 4.0
  - A division by zero in one of the application programs brought down the operating system.
  - NT was considered not responsible, since it was the application program that had malfunctioned.
  - Reference: <http://lists.essential.org/1998/am-info/msg03829.html>
- **We would prefer to use an OS that satisfied the above security requirement.**

## Common Criteria / Common Evaluation Methodology

---

- **Security concern for an Information Technology (IT) product**
  - Protect *assets* from threats that may compromise *confidentiality, integrity, and/or availability*
- **International standards for IT security**
  - ISO/IEC 15408, Parts 1-3: Criteria for IT Security Evaluation
    - Part 1: Introduction and General Model
    - Part 2: Security Functional Requirements
    - Part 3: Security Assurance Requirements
  - ISO/IEC 18045: Common Methodology for IT Security Evaluation

## Common Criteria / Common Evaluation Methodology

---

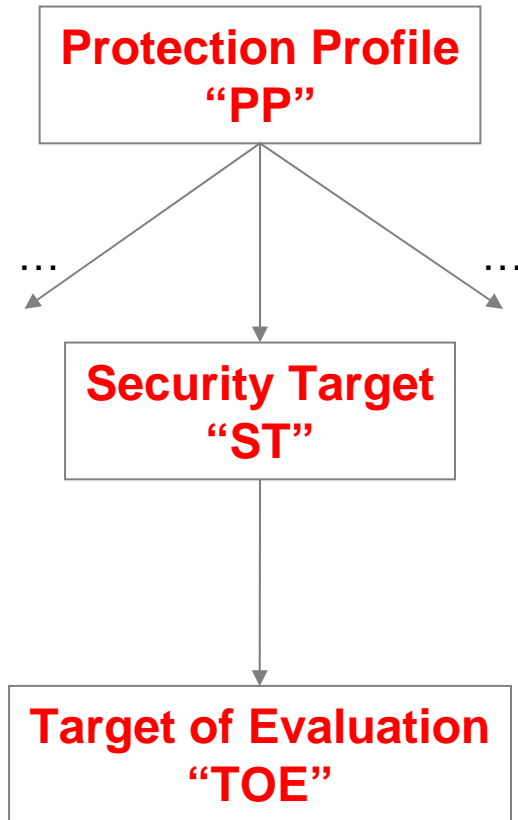
- **Purpose**

- “... provide a complete methodology for specifying IT security requirements, designing a solution to meet those requirements, and conducting an independent evaluation to ensure that all security requirements have been implemented ... correctly.”  
[Herrmann, p. 54]
- Usable by customers, developers, evaluators

- **Status**

- Current version is 3.1 Revision 2 (September 2007)

# Common Criteria Artifacts



## Consumer-specified implementation-independent specification of security requirements:

- Identify / characterize assets to be protected
- Security functional requirements (SFRs)
- Security assurance requirements (SARs) ⇒ Evaluation Assurance Level (EAL)

## Vendor-supplied implementation-dependent design based on a PP

- Specifies security mechanisms to meet PP requirements
- Basis for developing a Target of Evaluation
- One PP may generate many STs

## Vendor-supplied implementation of a ST

- "IT product, system or network and its associated administrator and user guidance documentation that is the subject of an evaluation" [Herrmann, p. 14]
- What gets evaluated are principally the TOE's Security Functions (TSFs), which are the implementation of the Security Functional Requirements specified in the PP
- Evaluation conducted by an accredited Common Criteria Testing Lab, reviewed by National Evaluation Authority

## Evaluation Assurance Levels (EALs)

Level	Description	Assurance
<b>EAL 1</b>	<b>Functionally tested</b>	<b>Low</b>
<b>EAL 2</b>	<b>Structurally tested</b>	
<b>EAL 3</b>	<b>Methodically tested and checked</b>	
<b>EAL 4</b>	<b>Methodically designed, tested, and reviewed</b>	<b>Medium</b>
<b>EAL 5</b>	<b>Semiformally designed and tested</b>	
<b>EAL 6</b>	<b>Semiformally verified design and tested</b>	
<b>EAL 7</b>	<b>Formally verified design and tested</b>	<b>Highest</b>

## EAL Levels and DO-178B Safety Levels

- **Charts seem similar in nature**
  - But they are VERY different
- **You can say “My program meets the requirements of DO-178B Level-A”**
  - Might not be true of course, but it is a meaningful statement
- **You cannot say “My program meets the requirements of EAL-7”**
  - This on its own is a meaningless statement
  - More than one OS vendor made such meaningless statements!
- **You can only talk about an EAL level in conjunction with a particular Protection Profile.**
  - Each PP is associated with an EAL level
  - Right now there are no defined PP's at level 7

## A Striking Example

---

- Various versions of Microsoft Windows (e.g. Windows 2000 at SP3) are certified at EAL4
- EAL4 is a pretty high level, in some respects equivalent to DO-178B Level A (the best you can do without formal methods).
- How can this be?
- Everyone who uses Windows knows that it is far from a totally reliable system.
- I certainly don't want the plane I am flying on to be running any version of Windows.
- Every week we read of hacks that can completely compromise a windows-based system.

## The Answer is in the Protection Profile

- **The applicable PP is the Controlled Access Protection Profile, Version 1.d (“CAPP”)**
  - [niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp\\_os\\_ca\\_v1.d/](http://niap-ccevs.org/cc-scheme/pp/pp.cfm/id/pp_os_ca_v1.d/)
  - “The CAPP provides for a level of protection, which is appropriate for an **assumed non-hostile and well-managed user community** requiring protection against threats of **inadvertent or casual attempts to breach the system security.**”
  - “The profile is **not intended to be applicable** to circumstances in which protection is required against **determined attempts by hostile and well-funded attackers** to breach system security.”
  - “The CAPP does not fully address the threats posed by malicious system development or administrative personnel.”
- **Bottom Line**
  - To understand the implications of a product’s EAL, read the Protection Profile!

# The Role of Testing

---

- **Safety-Critical Certification has mostly been based on testing.**
  - Not a guarantee of anything
  - No theoretical basis for claims of safety can be made from testing
  - But in practice works fairly well
  - No lives lost from software bugs on commercial flights
  - Though there have been some close calls
    - Malaysian airlines 777 incident
    - <http://chess.eecs.berkeley.edu/hcssas/papers/Shah%20Avionics%202.0%20doc.pdf>
- **But testing can never prove security**
  - That's why high EAL levels require increasing levels of formal reasoning. You have to *prove* that a security property is met.

## Relation Between Safety and Security

- **At a high level, Safety => Security**
  - Any safety-critical program these days has security aspects. If a failure of a program can cause loss of life, then a successful intrusion and compromise of the system has the same potential.
  - In an episode of ALIAS, Marshall hacks into the avionics system of a plane he is riding in. Amusing fiction? Or worrisome possibility?
  - Entertainment network on Boeing 787 is interconnected with network used for flight control. Security risk?
  - Historically less thought given to such considerations
  - Clearly safety-critical programs are a target of cyber-terrorism
- **So we have to consider Security Aspects for any Safety-Critical Program.**

## The Other Direction?

---

- **Are security-critical programs necessarily safety-critical?**
- **In the narrow sense, no, because we have defined safety-critical quite narrowly.**
- **But in practice, many programs while not safety-critical, are none the less very critical.**
  - An out of range subscript in a COBOL program caused Chemical bank to borrow billions from the Fed overnight, causing a stock market “adjustment”, affecting millions.
  - An error in Moody’s rating program caused incorrect assignment of grade A to risky real-estate investment vehicles.
  - Many websites have been compromised causing identity-theft affecting large numbers of people.
  - Voting machine errors could have catastrophic consequences

## Technical Convergence

---

- **Historically the safety and security communities have been rather distinct.**
- **But, as we found out in the Ada 95 development effort, they have similar technical concerns:**
  - Tools that minimize programming errors
  - Programming language features that minimize errors
  - Tools and languages that facilitate formal reasoning
  - For example, the SPARK dialect of Ada
  - We ended up with one Annex covering both concerns
- **DO-178B level A is equivalent to EAL-4**
- **For higher EAL levels, we need formal methods**
  - But many argue these days that relying on testing is inadequate even for safety-critical purposes

## Formal Methods and Safety, An Example

---

- **Everyone wants to use Object-Oriented techniques in safety-critical programs.**
- **But the notion of thorough testing and coverage testing is not a happy companion to OOT**
  - OOT are about obscuring details of control paths
  - Coverage testing is about knowing all possible control paths
- **One approach considered for DO 178-C**
  - Require Liskov Substitutability Principle to be followed
  - This basically says subclasses behave like the parent class in some reasonable technical sense in terms of pre/post conditions.
  - Then we know that any dispatching call will “work”.
  - But how do we show that LSP is met
  - Difficult without formal methods

## How are we doing now – Part 1 - Safety

- **As mentioned, for safety-critical programs, not so badly in cases where we follow standards carefully.**
  - But there are many areas where we don't follow standards so carefully (automotive applications, medical applications, power distribution systems, chemical plant control, ...)
  - There is an increased emphasis on the use of formal methods
  - But industry is nervous of costs (and prefers tools like static analysis, believing they can find bugs automatically).
  - Still, the costs of failure are unimaginably high in some cases
  - We are improving our techniques for formal reasoning
  - There are good examples of the use of formal methods in practice
    - See for example the Praxis work
    - Industrial Strength Exception Freedom
    - [http://www.praxis-his.com/pdfs/Industrial\\_strength.pdf](http://www.praxis-his.com/pdfs/Industrial_strength.pdf)

## How are we doing now – Part 1 - Security

---

- **Really not so well**
- **Every week, there are headline stories involving compromise of computer systems**
- **This week (really last week's) selection**
- **Palin's email being hacked**
  - Due to badly designed email security systems at yahoo, combined with naïve use of these features.
- **Large Hadron Collider Website hacked**
  - And had to be taken off line
  - It will be interesting to see if there is any software component to the massive failure of the collider this week.

# The Future of Critical Software

---

- **More and more, we rely entirely on critical software in our daily lives.**
  - Your car could crash
  - Your bank account could disappear
  - You could end up on some terrorism watch list
  - Your credit record could be destroyed
  - The plane you are on could crash
  - A chemical plant could blow up (think Bhopal \* 100)
  - Power could fail on a massive scale
  - A major refinery could fail, causing petrol prices to go through the roof (well perhaps not a good enough metaphor at this stage)
  - Voting machines could elect the wrong person
  - Cyber-terrorists could take control of the financial system
  - Etc etc etc

## We are starting to think seriously!

---

- **This conference and workshop illustrate the fact that more and more people are concerned with these issues.**
- **The US Department of Homeland Security has some excellent initiatives in this area.**
  - But try to find the pointers to them on the DHS home page
  - Not so easy, see [http://www.theregister.co.uk/2006/02/13/us\\_cyber\\_storm/](http://www.theregister.co.uk/2006/02/13/us_cyber_storm/)
- **Much more interest in this area at universities**
- **Safety and Security communities engaging**
  - For example at the annual NSA conferences in Washington
- **In our own area we see increased interest in tools**
  - Revisiting secure programming languages (Ada, SPARK)

## Conclusion

---

- We need to continue to develop technologies that allow the construction of large scale completely reliable programs.
- We need to build a security culture, in which aspects of security are considered at every stage.
- These technologies need to be widely used and disseminated and taught.
- People worry about the additional costs that may be imposed on software development.
- But in the long run, the cost of failures can be far higher (consider the Ebay example!)