SYSTEM EVALUATION AND ASSURANCE

Almog Apirion

Security Engineering – Chapter 26
Intro and definitions

- **Assurance** - whether the system will work
  - fundamentally comes down to the question of whether capable motivated people have beat up on the system enough

- **Evaluation** - how you convince other people of this
  - how you convince your boss, your clients, and in extremis, a jury that the system does indeed work (or that it did work in the past)

- How do you make a decision to ship the product?
- How do you sell the safety case to your insurers?
- How do you deal with people who protect the wrong thing, because their model of the requirements is out-of-date or plain wrong? And how do you allow for human failures?
- **Cost of protection and the risk of failure**
The players, playground and incentives

- **The vendor** would prefer that bugs weren’t found, to spare the expense of patching.

- **The average customer** might prefer the same; lazy customers often don’t patch, and get infected as a result. (So long as their ISP doesn’t cut them off for sending spam, they may not notice or care.)

- **The typical security researcher** wants a responsible means of disclosing his discoveries, so he can give the vendors a reasonable period of time to ship a patch before he ships his conference paper; so he will typically send a report to a local computer emergency response team (**CERT**) which in turn will notify the vendor and publish the vulnerability after 45 days.
The players, playground and incentives

- **The intelligence agencies** want to learn of vulnerabilities quickly, so that they can be exploited until a patch is shipped.

- **Hackers** disclose vulnerabilities on mailing lists such as bugtraq which don’t impose a delay; this can force software vendors to ship emergency patches out of the usual cycle.

- **The security software companies** benefit from the existence of unpatched vulnerabilities in that they can use their firewalls to filter for attacks using them, and the anti-virus software on their customers’ PCs can often try to intercept such attacks too.

- **Large companies** don’t like emergency patches, and neither do most government departments, as the process of testing a new patch against the enterprise’s critical systems and rolling it out is expensive.
Assurance

- The likelihood that a system will fail
- This estimate is based on:
  - the process used to develop the system
  - the identity of the person or team who developed it
  - particular technical assessments:
    - the use of formal methods or the deliberate introduction of a number of bugs to see how many of them are caught by the testing team; and experience
    - What will happened over time as a system is subjected to testing (new versions), use and maintenance
The things of which we need assurance

- **Incentives**
  - critical
  - If **people** don’t actually want to protect a system it’s hard to make them
  - **people** are the most critical part of the environment within which the security policy has to be defined

- **Policy**
  - often neglected
  - people often end up protecting the wrong things, or protecting the right things in the wrong way.
  - how you’d go about developing a policy for a new application.
The things of which we need assurance

- **Mechanisms**
  - *U.S. export controls on crypto* led to products like DVD being shipped with 40-bit keys that were intrinsically vulnerable
  - Strength of mechanisms is independent of policy, but can interact with it

- **Implementation**
  - Assurance traditionally focused on and was about whether, given the agreed functionality and strength of mechanisms, the product has been implemented correctly.
  - *most real life technical security failures are due to programming bugs* (stack overflows, race conditions, …) Finding and fixing them absorbs most of the effort of the assurance community.
The things of which we need assurance

- The big missing factor in the traditional approach to evaluation is **Usability**
  - Most system-level (as opposed to purely technical) failures have a significant human component.
  - Designers often see assurance simply as an absence of obvious bugs, and tie up the technical protection mechanisms without stopping to consider human frailty.
  - Example: access controls provided with operating systems often aren’t used, as it’s so much simpler to make code run with administrator privilege.
Usability

- **User Account Control**
  
  *user account may have administrator privileges assigned to it, but applications that the user runs do not inherit those privileges unless they are approved beforehand or the user explicitly authorizes it*
Security Testing

In practice, security testing usually comes down to reading the product documentation, then reviewing the code, and then performing a number of Tests (white box).

Assurance as a process is very much like the development of code. Just as you will have bugs in your code, you will also have bugs in your test procedures.
Security Testing

- **Look for any architectural flaws**
  - Does the system use guessable or too-persistent session identifiers?
  - Is there any way you can inject code, for example by sneaking SQL through a web server into a back-end database?
  - Where you can do wicked things?

- **Look for implementation flaws**
  - Stack overflows and integer overflows. This will usually involve not just looking at the code, but using specialist tools.

- **Then work down a list of less common flaws**
  - If the product uses crypto, look for weak keys and poor random-number generators.
  - If it has components with different trust assumptions, try to manipulate the APIs between them, looking for race conditions …
Quis Custodiet?

- Just as mistakes can be made by theorem provers and by testers, so they can also be made by people who draw up checklists of things for the testers to test. This is the old problem of *quis custodiet ipsos custodes*—who shall watch the watchmen?

- **Fault injection**
  - *Number of errors are deliberately introduced into the code at random.*

  If there are a hundred such errors, and the tester finds seventy of them plus a further seventy that weren’t deliberately introduced, then once the thirty remaining deliberate errors are removed you might expect that there are thirty bugs left that you don’t know about.
Process Assurance

- process measures such as who developed the system
- whether people are responsible for correcting their own bugs
  - waterfall model

- one team wrote the specification, another wrote the code, yet another did the testing (including some bug fixing), while yet another did the maintenance (including the rest of the bug fixing)
Microsoft - ‘if you wrote it, you fix it’

Bugs should be fixed as soon as possible, and even though they’re as inevitable as death and taxes, programmers should never give up trying to write clean code.

For years, internal auditors have included process issues in evaluating the quality of security code.

- hard because the organization’s quality culture is intangible. While some rules seem to be fairly universal, imposing a large number of specific rules would induce a bureaucratic culture rather than a dynamic competitive one.
Process Assurance

- Capability Maturity Model (CMM)
  - process improvement approach
  
  - CMM is based on the idea that competence is a function of teams rather than just individual developers
  - Problem: firms are forever reorganizing

- ISO 9001
  - A company must document its processes

23-Dec-12
Assurance Growth

- Assurance level of evolving products - Reliable growth
  - Grow product keep quality

- New releases mean new bugs, mean new security issues

- New products mean many new patches
  - Poisson distribution: $p = e^{-Et}$
    - probability $p$ that the bug remains undetected after $t$ statistically random tests
    - $E$ depends on the proportion of possible inputs that it affects
  - mean time between failure (MTBF)
    - ‘If you want a mean time between failure of a million hours, then you have to test for (at least) a million hours’
  - ‘Murphy’s Law
    - the number of defects that survive a selection process is maximized
example. Suppose a complex product such as Windows Vista has 1,000,000 bugs each with an MTBF of 1,000,000,000 hours

Suppose that Ahmed works in a cave where his job is to break into the U.S. Army’s network to get the list of informers in Baghdad

Brian is the army assurance guy whose job is to stop Ahmed.

- he must learn of the bugs before Ahmed does

Ahmed can only do 1,000 hours of testing a year

Brian has all the resources he can possibly dream of:

- full Vista source code, dozens of PhDs, control of the commercial evaluation labs, an inside track on CERT, an information sharing deal with any firm or government and consultants

- Brian does 10,000,000 hours of testing a year
After a year, Ahmed finds a bug, while Brian has found 10,000
- But the probability that Brian has found Ahmed’s bug is only 1%
- Even if Brian gets 100,000,000 hours of testing done each year
  - After ten years he will find Ahmed’s bug.
  - But by then Ahmed will have found nine more, and it’s unlikely that Brian will know of them

Conclusion, in critical systems and infrastructure you can’t just rely on a large complex commercial off-the-shelf product.

- You have to have mandatory access controls, implemented in an environment simple enough to verify
- Simplicity is the key to escaping the statistical trap
Evaluation

- the process of assembling evidence that a system meets, or fails to meet, a prescribed assurance target

- The fundamental problem is the tension that arises when the party who implements the protection and the party who relies on it are different
  - Simple case: you design a burglar alarm to standards set by insurance and have it certified by inspectors at their laboratories
  - Almost every information security product is very complicated
    - multiple and opposite principals and interests are involved
    - ‘no-one ever got fired for buying IBM’

- It is convenient to break evaluation into two cases
  - the evaluation is performed by the relying party (NASA)
  - the evaluation is done by someone other than the relying party
The Orange Book

- the Orange Book - the Trusted Computer Systems Evaluation Criteria (TCSEC)
  - Developed by National Computer Security Center, US Dept. of Defense
  - 1983–1999
  - Series that expanded on Orange Book in specific areas was called Rainbow Series
The Orange Book

The Orange Book and its supporting documents set out a number of evaluation classes:

- **C1**: discretionary access control by groups of users
  - In effect, this is considered to be equal to no protection.

- **C2**: discretionary access control by single users
  - Object reuse; audit
  - C2 corresponds to carefully configured commercial systems; for example, C2 evaluations were given to IBM mainframe operating systems, and to Windows NT.
    - Both of these were conditional on a particular configuration, in NT’s case, for example, it was restricted to diskless workstations.

- **B1**: mandatory access control
  - All objects carry security labels and the security policy is enforced independently of user actions. Labeling is enforced for all input information.

- **B2**: structured protection
  - As B1 but there must also be a formal model of the security policy that has been proved consistent with security axioms.
  - Tools must be provided for system administration and configuration management. The TCB must be properly structured and its interface clearly defined. Covert channel analysis must be performed. A trusted path must be provided from the user to the TCB. Severe testing, including penetration testing, must be carried out.

- **B3**: security domains
  - As B2 but the TCB must be minimal, it must mediate all access requests, it must be tamper-resistant, and it must withstand formal analysis and testing.
  - There must be real-time monitoring and alerting mechanisms, and structured techniques must be used in implementation.

- **A1**: verification design
  - As B3, but formal techniques must be used to prove the equivalence between the TCB specification and the security policy model.
- functionality is multi-dimensional
- assurance has a linear progression
The orange book disadvantages

- The process was driven and controlled by the Government, the party that was going to rely on the results of the evaluation, while the vendor was the supplicant at the gate.
- Because of the time the process took, evaluated products were usually one or two generations behind current commercial products.
- The incentive issues were not properly thought through.
- The Orange Book wasn’t making procurement easy and contractors detested having to obtain separate evaluations for their products.
- Expensive and not flexible.
The orange book Contributions

- Heightened awareness in commercial sector to computer security needs
- Led to wave of new approaches to evaluation
  - As commercial firms could not use it for their products, some commercial firms began offering certifications
- Basis for several other schemes, such as Federal Criteria, Common Criteria
The Common Criteria

- An international standard (ISO/IEC 15408)
- Began in 1998 with signing of Common Criteria Recognition Agreement with 5 signers
  - US, UK, Canada, France, Germany
- As of May 2002, 10 more signers
  - Australia, Finland, Greece, Israel, Italy, Netherlands, New Zealand, Norway, Spain, Sweden; India, Japan, Russia, South Korea developing appropriate schemes
- Vendors pays for evaluation of their products
- Evaluation is made by contractors certified as a commercial licensed evaluation facility (CLEF)
The Common Criteria

- The Common Criteria vs. the orange book
  - the Common Criteria have much more flexibility than the Orange Book
  - Rather than expecting all systems to conform to Bell-LaPadula, a product is evaluated against a protection profile
  - There are protection profiles for operating systems, access control systems, boundary control devices, intrusion detection systems, smartcards, key management systems, VPN clients...
  - The tent is certainly a lot broader than with the Orange Book
  - However, anyone can propose a protection profile
- Does not provide one list of security features
- Describes a framework where security requirements can be specified, claimed, and evaluated
The Common Criteria Key concepts

- **Target Of Evaluation (TOE)**
  - the product or system that is the subject of the evaluation.

- **Protection Profile (PP)**
  - a document that identifies security requirements relevant to a user community for a particular purpose.

- **Security Target (ST)**
  - a document that identifies the security properties one wants to evaluate against

- **Evaluation Assurance Level (EAL)**
  - a numerical rating (1-7) reflecting the assurance requirements fulfilled during the evaluation
CC Assurance Requirements

- Ten security assurance classes

Classes:
- Protection Profile Evaluation
- Security Target Evaluation
- Configuration Management
- Delivery and Operation
- Development
- Guidance Documentation
- Life Cycle
- Tests
- Vulnerabilities Assessment
- Maintenance of Assurance
Protection Profiles (PP)

- protection profile (PP) is an implementation-independent set of security requirements for a category of products or systems that meet specific consumer needs
  - Subject to review and certified

- Requirements
  - Functional
  - Assurance
  - EAL
Protection Profiles

- Example: Controlled Access PP (CAPP_V1.d)
  - Security functional requirements
    - Authentication, User Data Protection, Prevent Audit Loss
  - Security assurance requirements
    - Security testing, Admin guidance, Life-cycle support, ...
  - Assumes non-hostile and well-managed users
  - Does not consider malicious system developers
Security Targets (ST)

- “A security target (ST) is a set of security requirements and specifications to be used for evaluation of an identified product or system”
- Can be based on a PP or directly taking components from CC
- Describes specific security functions and mechanisms
Evaluation Assurance Levels 1 – 4

EAL 1: Functionally Tested
- Review of functional and interface specifications
- Some independent testing

EAL 2: Structurally Tested
- Analysis of security functions, incl. high-level design
- Independent testing, review of developer testing

EAL 3: Methodically Tested and Checked
- More testing, Some dev. environment controls;

EAL 4: Methodically Designed, Tested, Reviewed
- Requires more design description, improved confidence that TOE will not be tampered
Evaluation Assurance Levels 5 – 7

EAL 5: Semiformally Designed and Tested
- Formal model, modular design
- Vulnerability search, covert channel analysis

EAL 6: Semiformally Verified Design and Tested
- Structured development process

EAL 7: Formally Verified Design and Tested
- Formal presentation of functional specification
- Product or system design must be simple
- Independent confirmation of developer tests
CC catalogues

- There are catalogues of:
  - threats, such as T. Load Mal- Data loading malfunction:
    - an attacker may maliciously generate errors in set-up data to compromise the security functions of the TOE
  - assumptions, such as A. Role Man- Role management:
    - management of roles for the TOE is performed in a secure manner (in other words, the developers, operators and so on behave themselves)
  - organizational policies, such as P. Crypt Std- Cryptographic standards:
    - cryptographic entities, data authentication, and approval functions must be in accordance with ISO and associated industry or organizational standards
  - objectives, such as O. Flt Ins- Fault insertion:
    - the TOE must be resistant to repeated probing through insertion of erroneous data
  - assurance requirements, such as ADO DEL.2- Detection of modification:
    - the developer shall document procedures for delivery of the TOE or parts of it to the user
it’s not the nominal CC level that tells you anything, but the details of the PP.

the first question you should ask when told that some product has a Common Criteria Evaluation is: ‘against what protection profile?’

‘the fact that an IT product has been evaluated has meaning only in the context of the security properties that were evaluated and the evaluation methods that were used’

So who’s actually liable?
The Common Criteria

- it’s not the nominal CC level that tells you anything, but the details of the PP.
  - the first question you should ask when told that some product has a Common Criteria Evaluation is: ‘against what protection profile?’
  - ‘the fact that an IT product has been evaluated has meaning only in the context of the security properties that were evaluated and the evaluation methods that were used’
- So who’s actually liable?
<table>
<thead>
<tr>
<th>Product</th>
<th>EAL</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>VMware® ESXi Server 3.5 and VirtualCenter 2.5</td>
<td>EAL4+</td>
<td>24-FEB-10</td>
</tr>
<tr>
<td>Microsoft Windows Mobile 6.5</td>
<td>EAL4+</td>
<td>09-FEB-10</td>
</tr>
<tr>
<td>Apple Mac OS X 10.6</td>
<td>EAL3+</td>
<td>08-JAN-10</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux Ver. 5.3 on Dell 11G Family Servers</td>
<td>EAL4+</td>
<td>23-DEC-09</td>
</tr>
<tr>
<td>Windows Vista Enterprise; Windows Server 2008 Standard Edition;</td>
<td>EAL4+</td>
<td>31-AUG-09</td>
</tr>
<tr>
<td>Oracle Enterprise Linux Version 5 Update 1</td>
<td>EAL4+</td>
<td>15-OCT-08</td>
</tr>
</tbody>
</table>
Common Criteria criticism

- High cost
- Bureaucracy
- too focused on the technical aspects of design:
  - things like usability are almost ignored
  - firm’s administrative procedures with the technical controls is outside the scope
- don’t cope well with change
  - operating systems such as Windows and Linux have been evaluated, but in very restricted configurations (typically, a workstation with no network connection or removable media- where all the evaluation is saying is that there’s a logon process and that filesystem access controls work).
- Products with updates
- Ignoring the business processes
  - Technical mechanisms shouldn’t be used where the exposure is less than the cost of controlling it, or where procedural controls are cheaper
Common Criteria evaluations are done by CLEFS contractors who are licensed by the local signals intelligence agency and there for influenced by national interests.

- the vendor pays the CLEF
  - can shop around for a CLEF that will give it an easy ride technically, or that will follow its political line
  - No CLEF license had been revoked
A more realistic approach to evaluation and assurance would look not just at the technical features of the product but at how it behaves in real use.

In most applications, one must assume that people are always careless, usually incompetent and occasionally Dishonest.

large, feature-rich programs that are updated frequently. Economics cannot be wished away. Evaluation and assurance schemes such as the Common Criteria, ISO9001 and even CMM try to squeeze a very volatile and competitive industry into a bureaucratic straightjacket, in order to provide purchasers with the illusion of stability. But the establishment and maintenance of these brands involves huge market forces, and security plays little role.
Education

- problems and technologies of system protection need to be much more widely understood
  - the wrong mechanisms were used, or the right mechanisms were used in the wrong way
- Security professionals tend to be either too specialized and focused on some tiny aspect of the technology, or else generalists who’ve never been exposed to many of the deeper technical issues
Sometimes the hardest part of a security engineering project is knowing when you’re done.

A number of evaluation and assurance methodologies are available to help:
- In moderation they can be very useful.
- But the assistance they can give has its limits, and overuse of bureaucratic quality control tools can do grave harm.
- One can think of them as like salt; a few shakes on your fries can be a good thing, but a few ounces definitely aren’t.

People gradually acquire experience of what works, what gets attacked and how, and as protection requirements and mechanisms become more part of the working engineer’s skill set, things gradually get better.

Security may only be got right at the fourth pass, but that’s better than never—which was typical fifteen years ago.

Life is complex. Success means coping with it. Complaining too much about it is the path to failure.