Evidence (of what?) Based Security Assessment

Ed Reed
Security Tzar
Novell, Inc.

Novell

Scientific vs. Engineering Methods

Scientific – Conjecture, Theorize, Hypothesis, Test

- Starts with a question and explores alternative answers

Engineering – Analyze, Specify, Design, Build, Test

- Starts with an objective and proceeds to a deliverable

Unfortunately:

All too often with software,
no one has a clue as to
what it is, what it does, what it is supposed to do,
what it is NOT supposed to do

Biggest Impediment to Progress Is...

Lack of consensus on just about everything "security"

- baseline security policy
- vendor (!) vs central (helpdesk) vs individual control
- enterprise vs consumer, parent vs child
- end-to-end vs point-to-point with proxy intermediaries
- performance!

Even the best best practices aren't good enough!

- one-size just won't fit all
- need a few characteristic profiles to tackle
  - departmental servers, enterprise data center servers
  - public interface servers, proxies, guards
  - networking (routing, authentication) infrastructure

So what "evidence" can you show?

How about starting with a description of the environment?

- Defines experimental context and assumptions
  - you need experimental "controls", right?

Will you test to see if some security objectives are met?

- what security policy? is the system responsible for all aspects of them, or is the environment responsible for some?
- should you test to see which ones aren't met?

Do you know what the thing is supposed to do?

- what does it touch? what does it need to work?

Do you know what the thing is NOT supposed to do?

- can you prove it won't? How?

Congratulations

Your experimental report sounds a lot like a Common Criteria evaluation

It may not be perfect, but it DOES provide an Evidence Based Assessment of a product

And if it doesn't answer the questions you're asking -

- "are there buffer overflows", "can you tell what it's doing", "does it transmit your key in the SSL packet headers", "does it store your secrets in plain sight"

Then you're not looking in the right places, or

You need to bake your questions into the requirements

High Assurance Comes with Knowledge

What do you want the system to do?

Is the system designed to do that? Only that?

- What else does it do that you don't want?

Minimalist system design

- everything is BOTH necessary AND sufficient
- nothing extra you don't know about

Modular, Layered, Understood, Well Defined Interfaces

The Evidence is in the documentation, the formal design, the detailed analysis, and the insistence that nothing is there that isn't necessary

- the knowledge of what it does and doesn't do

Let's Start

Make it a requirement that software come with a manifest

- declaring all the files it loads, reads, modifies & executes
  - including version dependencies, please...
- identifying all the network ports it opens or listens on
  - and what it's doing with them!
- explaining all the command line arguments and showing examples that make sense
- telling you about all the configuration file tokens it thinks it understands, and what they do

Now you can build a test plan, and even inspect the code


Novell®