A Fault Model for Information Security Governance and its Uses as a Basis for Metrics

Fred Cohen
CEO - Fred Cohen & Associates
Research Professor - University of New Haven

Outline

- Background
  - Fred Cohen – What does he know about metrics?
  - Types of Metrics
- Some History
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

What does Fred know of metrics?

- About as much as anyone else here...
  - Studies and worked in testing and fault tolerant computing for a time and used those metrics
  - Did some design automation which uses a lot of metrics to make optimization decisions
  - Did a fair number of experiments in the early days of viruses and developed some limited metrics
  - Developed and used metrics to measure the effects of deception for information protection
  - Currently developing metrics for measuring enterprise protection programs
- Maybe even more than some... or not...

Types of Metrics

- From worst to best:
  - Nominal
    - Have user identities - Don't have authentication
  - Ordinal (produces a POSet)
    - Bad Publicity is worse than Getting a Virus
    - Death is worse than Failed Authentication
  - Interval (counting – but against what?)
    - My product catches 12,000 viruses, yours catches 6,000
    - (But yours catches the 6000 most common ones...)
  - Ratio (the facts are not right here)
    - Loss of one life costs $2.5M
    - Safety Belts cost $5M per life saved
    - Don't use Safety Belts!

Outline

- Background
- Some History
  - Fault Tolerant Computing and Testing
  - Fault Models, Coverage, and Other Such Stuff
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

FTC and Testing

- Digital circuit faults produce failures in outputs
  - If the faults are exercised
  - If they are not masked
- Testing improves quality
  - Provides the means to tell if circuits have certain kinds of faults before they become failures
- Fault Tolerance can compensate for faults
  - Provides redundancy for certain types of faults
  - But it fails spectacularly when it fails
- But it wasn't always that way

Fault models and coverage

- FTC found an epistemology
  - Fault models lead to finite sets of faults
    - Stuck at faults (for example)
    - Each input and output can be S-A-1 or S-A-0
    - Total wires * 2 = total possible faults
    - Assume single stuck-at-faults and analyze
  - Finite sets of faults lead to tests and coverage
    - Generate test vectors to “cover” all single stuck-at faults
    - Count the number covered / total possible -> “coverage”
  - Metrics come from coverage
    - Minimize the test vectors for a given coverage
    - Minimum vectors for 100% coverage
    - Test time for given coverage ... and on and on...

Other stuff

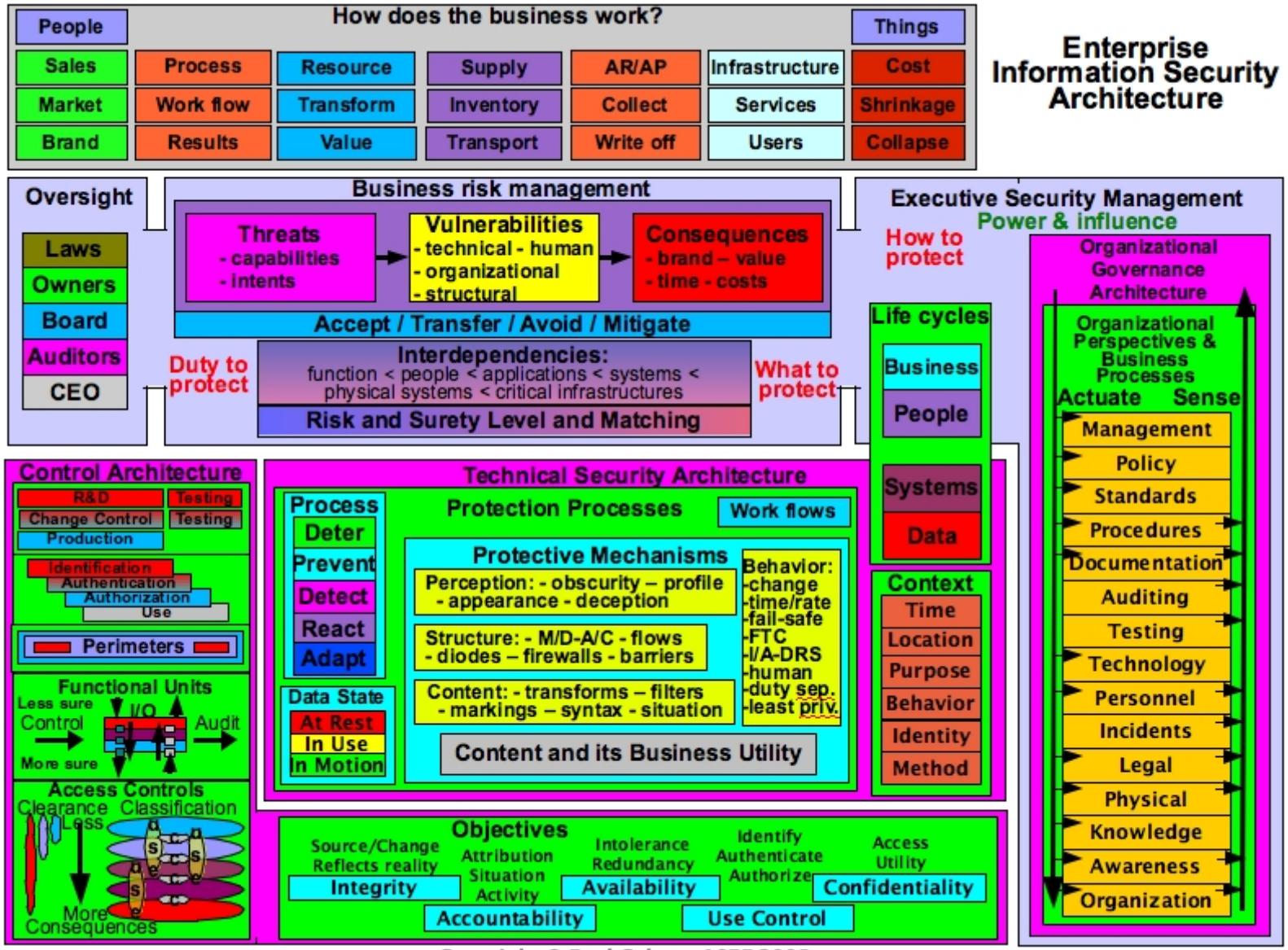
- But not all faults are equal!!! (and analogies)
  - One fault may be on a larger number of paths
    - More interdependencies in risk analysis
  - One fault may almost never be exercised (Intel)
    - Probability of occurrence is small
  - Redundancy may prevent link from fault to failure
    - I don't need gold plated security... surety levels
  - State machines may have many faults possible
    - Event sequences with potentially serious negative consequences
  - And on and on...
- Testing and FTC handled them all – over time

Outline

- Background
- Some History
- A Model of Security Governance
  - The Model and Where to Get it
  - Mapping the Model
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

Here it is...

- all.net -> Security Architecture



Why I use it

- I created it to use it for modeling governance
  - The governance guidebook
  - Security Metrics
  - Governance checklists
  - Software versions
  - Links to standards
- I don't have to pay royalties
  - You might – let's talk...
  - It's not particularly better than any other model, but it does have what I need to analyze governance issues for now.
  - Make your own model and do the same thing...

Mapping the model

- I have mapped it to ISO, ITIL, NIST, etc. - available on all.net as free downloads
- The other standards lack some things I want
  - They don't define a business model
  - They ignore duties to protect as a function
  - They ignore a level of detail on risk management
  - They don't recognize the cross cutting issues well
  - They ignore the control architecture as an entity
  - They don't map life cycles, process, context, data state, and seem to ignore work flows and process
  - They don't map protective mechanisms this way

Outline

- Background
- Some History
- A Model of Security Governance
- Measuring Against the Model
  - Taking Surveys and Challenges
  - What do Results Look Like
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

Surveys and Challenges

- How do I measure enterprise against model?
  - Used to do detailed visits and analysis – IPPA
  - Testing survey methodology – survey and verify
- C&S V25#6 ISO 17799 gap analysis paper
  - Claims surveys work for gap analysis for ISO
  - I beg to differ... bad data because:
    - People don't understand the issues as well as experts
    - People lie, fabricate, want to look good, are afraid
    - People are busy and don't want to take time and effort
    - People don't like top think
    - People...
- So I do surveys than analyze and verify/refute

What do results look like?

- During the demo period you can see more...
  - Basically, you ask about what's there and not there
  - Work your way around the model
  - Work your way up surety levels
  - Different representations of data are used to see different things
  - Here is some real data from a real survey – 21 max

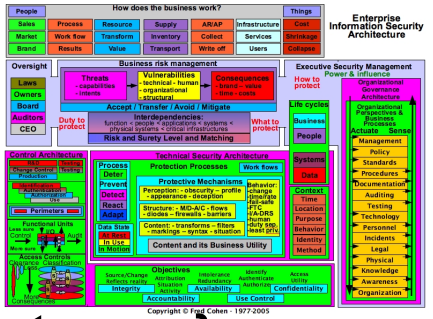
Mod		Duty	Mgt	Pol	Std	Proc	HR	Leg	RM	$s	Test	CC	Tech	Phy	Inc	Audit	Know	Aware	Doc	Mat
0	2	12	4	17	8	7	10	8	5	3	4	4	7	10	1	0	4	4	0	8
0	0	2	0	3	5	4	8	6	3	1	2	5	1	5	3	3	0	2	2	2
0	1	3	2	3	7	6	4	4	3	0	1	3	1	1	3	0	2	2	0	2
0	0	1	0	3	3	3	1	3	4	0	1	1	0	2	2	0	2	0	1	0
1	2	3	2	0	2	2	1	6	2	1	0	2	2	3	7	0	3	3	1	1

Security Performance Management Summary

Outline

- Background
- Some History
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
  - What does a Fault Model Look Like
  - The Fault Model we are Using
  - Model complexity
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

What does a fault model look like?



- Fault types:
  - Execution (touches business utility)
    - {**miss**, **make**, **mix**} do **action** on {**element**, **instance**}
  - Process
    - {**miss**, **make**, **mix**} **process** for {**element**, **instance**}
  - Management
    - {**miss**, **make**, **mix**} **control** over {**element**, **instance**}
  - Specification (or existence)
    - {**miss**, **make**, **mix**} **define** action on {**element**, **instance**}
- Causality:
  - Specification causes management causes process causes execution – faults go back up - sometimes

The fault model we are using

- Basically the one you just saw – but...
  - The devil is in the details...
    - Execution failed (configuration fault made authentication)
    - Was consequence above loss acceptance threshold?
      - No: Not risk management fault
        - Was worker properly trained and supervised?
          - No: Management fault + process fault
      - Don't know: risk management existence failure or missed risk
      - Yes: Process fault (process should prevent losses > threshold)
        - Miss / make / or inadequate?
          - Miss: Management fault
          - ...
- And there is another level of import
  - Correction means finding and fixing all relevant faults

Model complexity

- 2*4 = 8 individual fault types
  - {miss, make} x {define, control, process, act}
- On each of {element, instance} (pairs for causal links)
  - 6 top level elements = 48 fault types = 2256 pairs
  - 75 next-level elements = 600 fault types = 359400 pairs
- Links to hierarchical top-level elements
    - 21 TSA elements link to 5 higher level elements = 105
    - 20 CA elements link to 4 higher level elements = 80
    - 15 perspectives * 3 others = 45
    - 14 risk management * 2 others = 28
    - 5 oversight * 1 other = 5
  - Total to higher level causes = 263 * 8s * 8d types = 16832

Outline

- Background
- Some History
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
  - Measurements Lead to Characterizing Faults
  - Then there is Root Cause Analysis
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions - Discussion

Measure program or consequences

- Bad things happen – examine them - OR
- Take measurements using surveys
  - Proximate causes are pretty easy
  - Root causes require analysis
  - But statistics on responses or causes leads to...
- Characterize the faults with coverage
  - The previous table showed coverage levels
  - Divide each number by 21 and it gives the coverage
    - Surety increases with vertical so measure against risk
    - For higher risk you should have higher coverage

Root cause in the large

- Common characteristics of faults – an example
  - The survey has LOTS of faults (out of 21)
    - Why aren't they falling apart at the seams?
  - They are – they just don't know it!
    - As the number of faults goes up, the nature of the issues tend toward higher level faults as the common cause
    - They are operating at or below minimal diligence levels for low risk situations

Mod	Duty	Mgt	Pol	Std	Proc	HR	Leg	RM	$s	Test	CC	Tech	Phy	Inc	Audit	Know	Aware	Doc	Mat
0	2	12	4	17	8	7	10	8	5	3	4	4	7	10	1	0	4	4	0	8
0	0	2	0	3	5	4	8	6	3	1	2	5	1	5	3	3	0	2	2	2
0	1	3	2	3	7	6	4	4	3	0	1	3	1	1	3	0	2	2	0	2
0	0	1	0	3	3	3	1	3	4	0	1	1	0	2	2	0	2	0	1	0
1	2	3	2	0	2	2	1	6	2	1	0	2	2	3	7	0	3	3	1	1

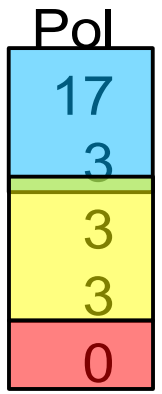
Security Performance Management Summary

Outline

- Background
- Some History
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
  - Coverage, Redundancy, and Uncertainty
  - Fault Models -> Root Causes -> Program Metrics
- Summary - Conclusions - Discussion

Coverage

- Coverage:
  - Faults covered / number of possible faults (in survey)
    - Fault = miss x {define, control, process, act} x element
  - Depending on surety level, different coverage is desired
    - e.g., Policy (assume medium risk -> medium surety)
    - 17/21 indicate awareness of legal issues: C=81%
    - 3/21 indicate any other policy coverage: C=14%
  - Coverage for whole or partial table is easy to calculate
    - Policy coverage relative to risk level is an example
    - Low risk, policy coverage = 20/42=48% (17+3/2*21)
    - Medium risk policy coverage = 26/84 = 31%
    - High risk policy coverage = 26/105 = 10%



Redundancy and Uncertainty

- Redundancy:
  - Just because you don't have policy doesn't mean you don't have effective protection in place
  - Examples of redundancy:
    - Awareness programs every quarter (over time)
    - R&D tests, change control tests (separation of duties)
    - Background checks and management controls (programatic)
- Uncertainty:
  - In this case, it's obvious...
    - Coverage is too low (175/882 = 20% coverage for low risk)
  - If it was close we would have to dig deeper...

Fault models -> Root causes

- The fault model says:
  - Existence faults lead to management faults lead to process faults lead to execution faults
  - When coverage of most of the management part of the model is in the range of C<10%:
    - Management miss leads to process miss
      - You should see lots of miss process
    - Lots of miss process means lots of miss execution
      - You should see lots of execution faults across many domains leading to losses
      - You should see many expensive temporary execution fixes instead of minor systemic changes to adapt
    - And that's exactly what we see (or saw in this case)
  - Lots of miss management means existence faults
    - Across enterprise means miss program existence

Root causes -> Program metrics

- So now we can:
  - Count faults (nominal) and get coverage (ratio)
  - Apply faults against a model (nominal)
  - Seek commonalities (ordinal)
  - Get root causes (nominal)
  - Define a strategy for improvement (repair faults)
  - Measure the program over time (faults -> coverage)
- Which I call program metrics

Outline

- Background
- Some History
- A Model of Security Governance
- Measuring Against the Model
- A Fault Model for the Governance Model
- Turning Measurements Into Root Causes
- Coverage + Fault Models = Program Metrics
- Summary - Conclusions – Discussion
  - Your Turn!

Summary and conclusions

- Follow the example of FTC, testing
  - Epistemology says we need a fault model
    - Use mine underlying model or build your own
    - Add fault sets and causality
    - Provide an analytical framework
  - Then start to define simple metrics
    - Coverage is an excellent start for governance
    - Expand them into other metrics as needed
  - Use the metrics with the model to get to causality
    - Causality leads to a model of repair process as well
  - Measure progress with the same model
    - Program metrics for mitigation and maintenance(t)

Thank You

all.net