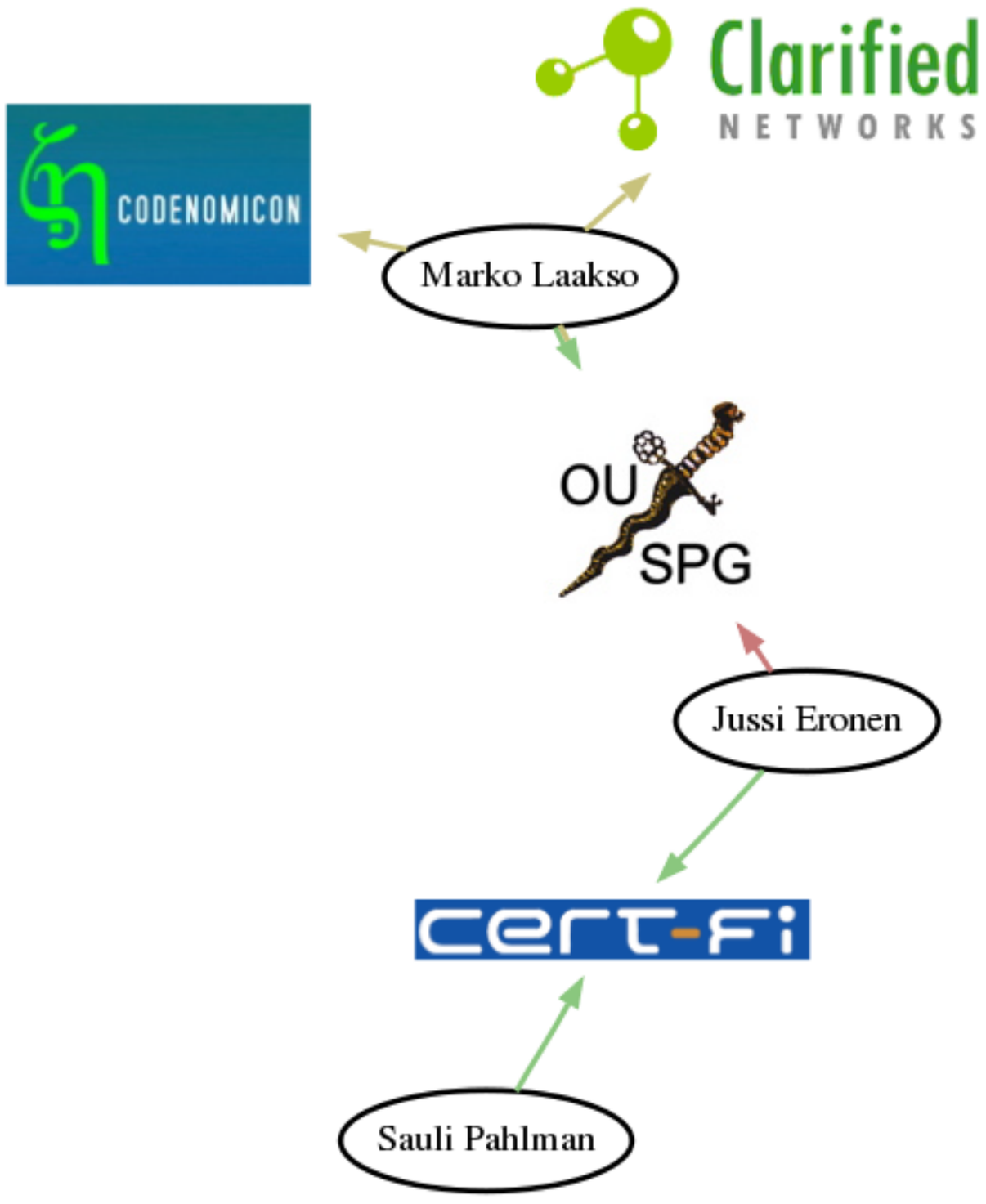
Setting the Scene in Vulnerability Disclosure



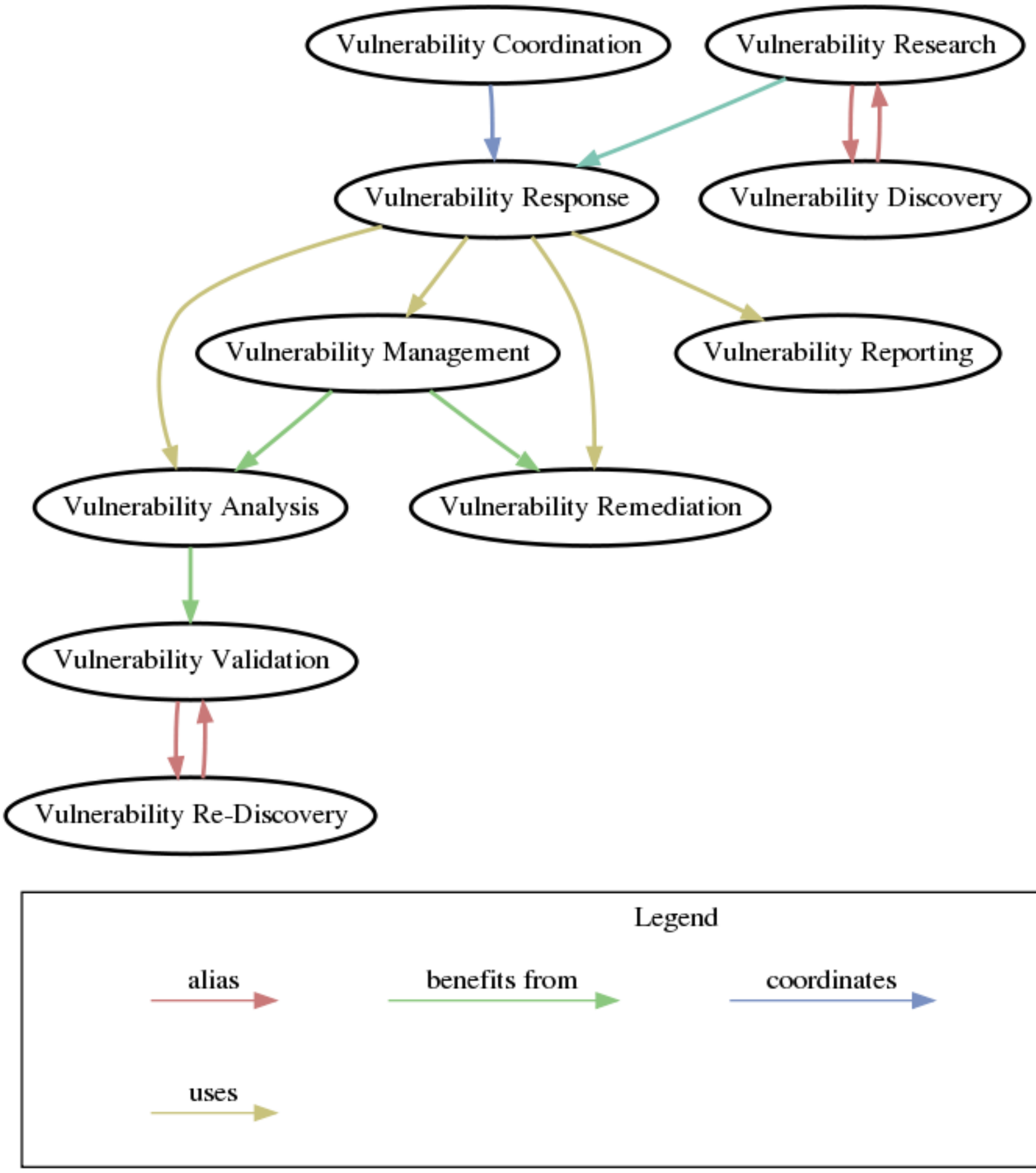
What is vulnerability discovery?

Who is doing it?

Why? What's in it for them?

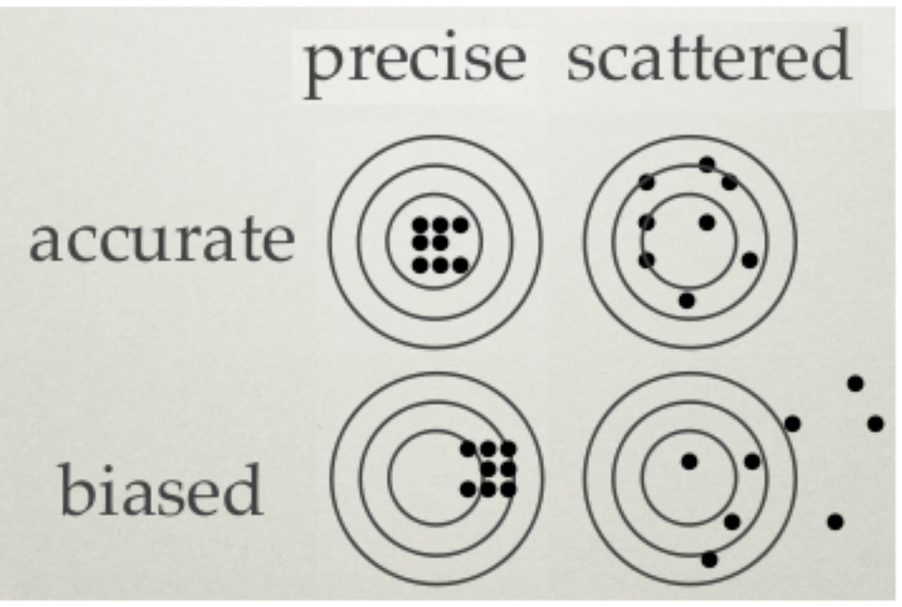
How is it being done?

Different Flavours of Vulnerability Work



What is vulnerability discovery?

- **Vulnerability** A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy (RFC2828)
  - **Vulnerability discovery**: The process of finding vulnerabilities
  - **Vulnerability rediscovery**: Analysing and reproducing a known vulnerability based on public or private data
  - **Coincidental discovery**: Finding a vulnerability that has been publically or privately reported, either accidentally or in the process vulnerability discovery
  - **Exploit**: script, program, mechanism, or other technique by which a vulnerability is used in the pursuit or achievement of some information assurance objective.
- For the purposes of this discussion, soft components of the systems are not considered
- Vulnerability discovery is a resource optimisation issue: **Cost of discovery vs benefit gained from discovering a vulnerability**
  - Or often: benefit of discovery vs benefit of unrelated actions
- Mostly iscussed as a deeply technical subject
  - Neglected fators: soft issues hindering or preventing discovery efforts, precision/accuracy



- Dan Geer, http://geer.tinho.net/measuringsecurity.tutorial.pdf

Why this work is done?

Motivations

«	Description «
Certification	Ensure that the product or systems meets predefined security and safety criteria
Competitive advantage	Use vulnerability information in offensive or defensive products to gain advantage over competitors without the information
Compliance	Compliance requirements stem from organisational policy requirements, commercial standards such as PCI, and legistlative requirements such as SOX, HIPAA and FISMA.
Liability	Liability for damages caused by the effects of software flaws and vulnerabilities
Offensive Capability	Find vulnerabilities to be able to use them in (real or simulated) attacks
Protect constituents	Protect clients or other constitients from the ill effects of vulnerabilities
Protect general public	Protecting the general public from the ill effects of vulnerabilities
Protect self	Protect your own organisation from the ill effects of vulnerabilities
Raise the bar	Try to improve the overall security level of products or systems
Reputation	Gain reputation by discovering vulnerabilities, Protect the damage to your reputation caused by vulnerabilities in your systems

Motivations - An Illustration

Reporters Receivers

- Tiina Havana, https://www.ee.oulu.fi/research/ouspg/PROTOS_Stanford2003
- In a 2002 study, it was identified that vulnerability reporters and report recipients somewhat agree on what is important in vulnerability disclosure

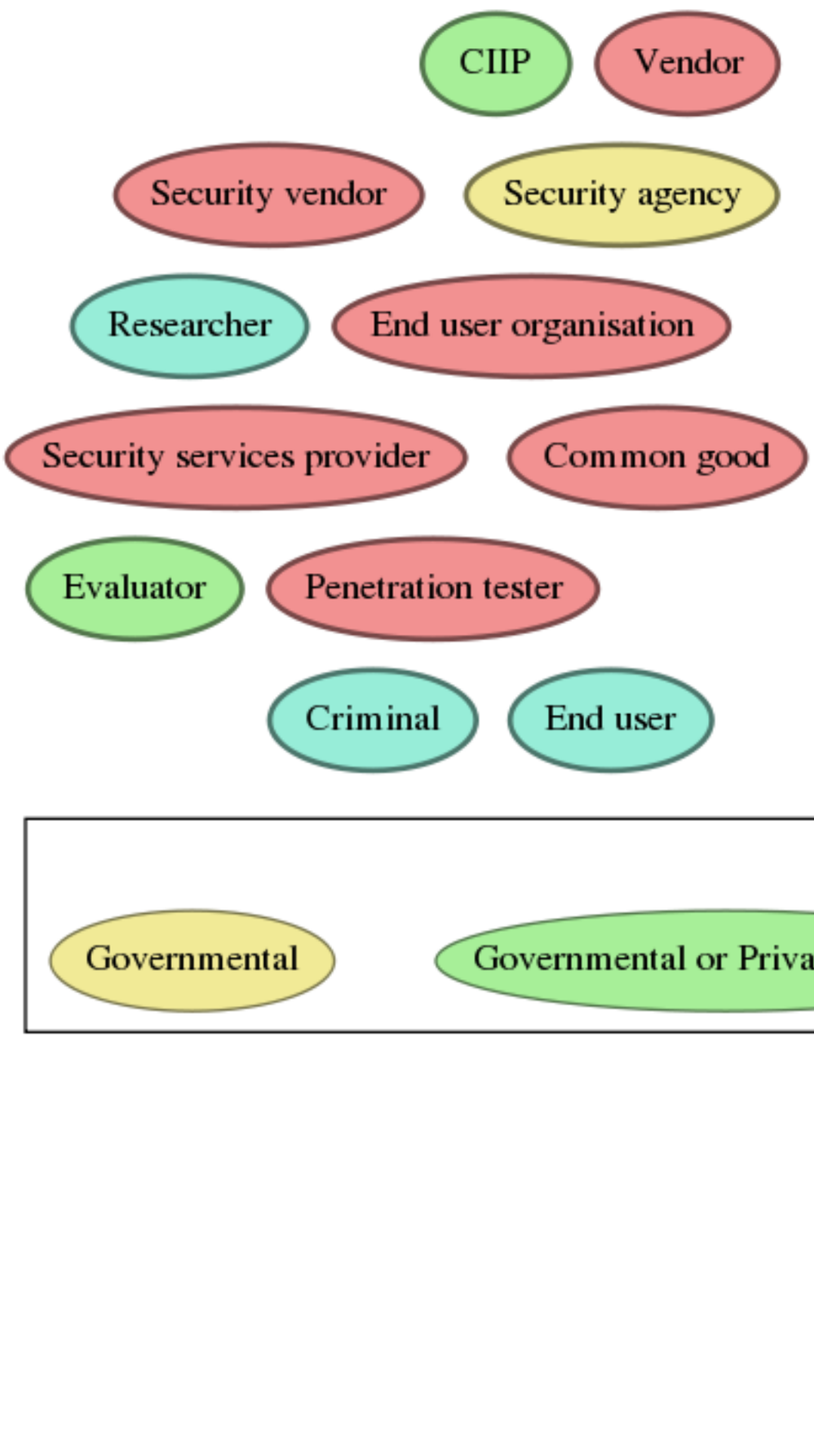
Issue/Importance	Reporter	Receiver
Security	1	1
Precision/accuracy	3	2
Non-maleficence	5	3

- This is why the process works at all
  - common disagreements: public's right to know, public benefit, FUD

Who is doing it?

Actors

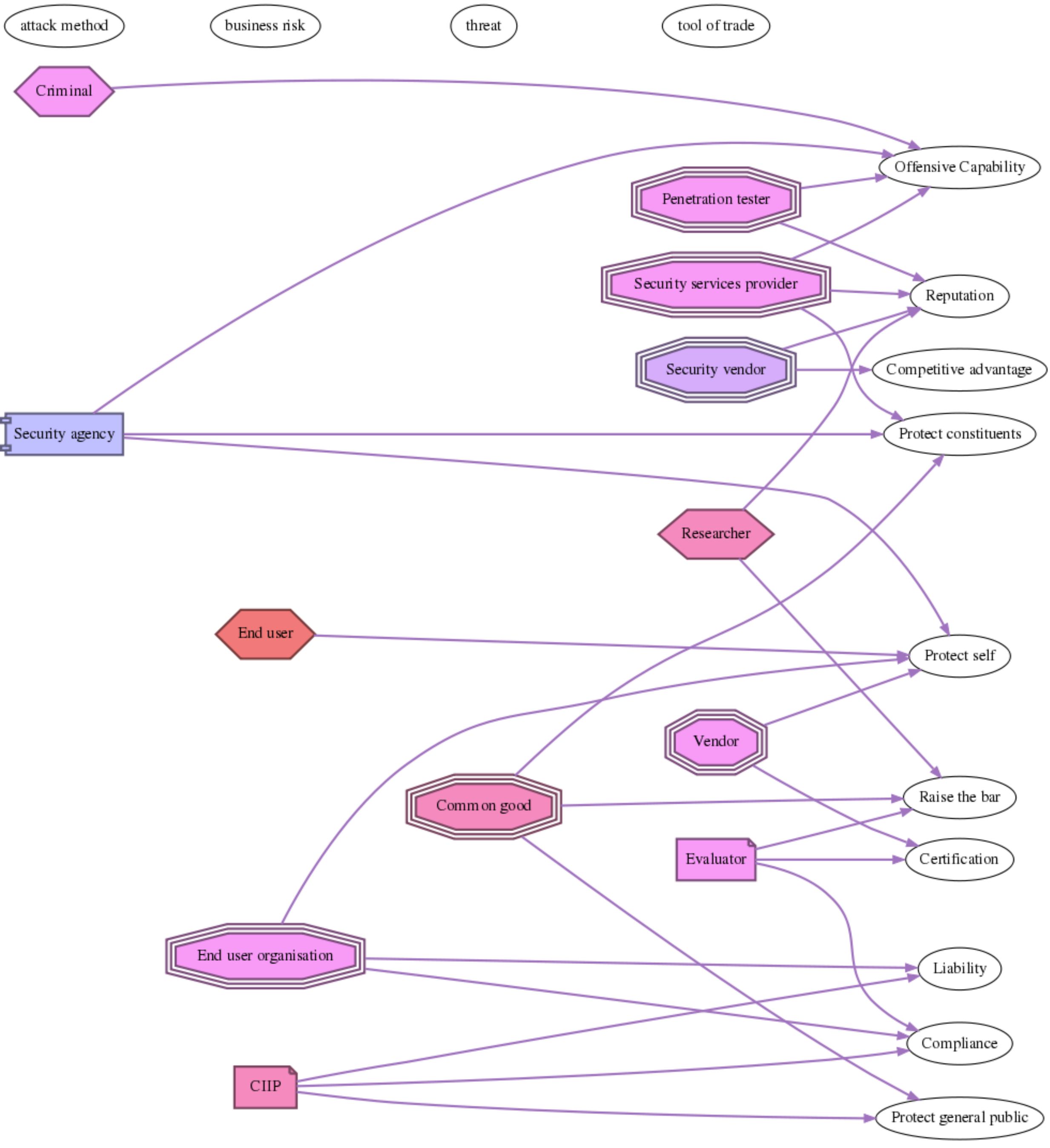
- Actors categorised on the following factors:
  - **range of available resources**: Ranges from poor to excellent
  - **motivation**: As discussed earlier
  - **disposition**: Is the actor governmental, privately owned, or an individual person(/persons)
  - **position** on vulnerabilities: Is it a weapon, a threat, an essential tool for their trade or a business risk
- A crude simplification, YMMV



Who is doing it?

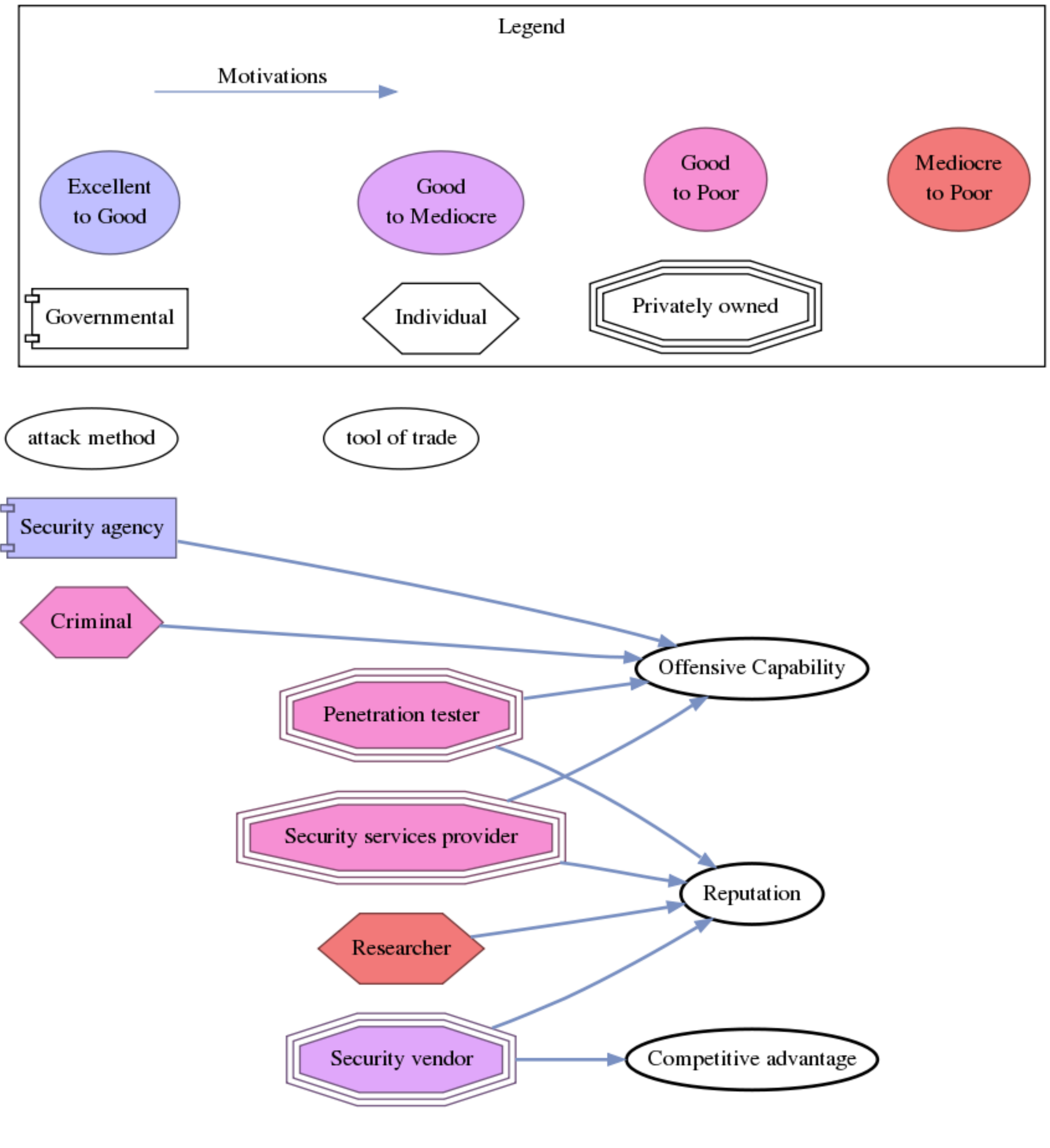
«	Description «	Resources «	Disposition «	Position «
Security agency	Actor from the sectors of governmental or military security	Excellent, Good	• Governmental	Attack method
Evaluator	Evaluates products or systems according to set criteria	Good, Poor	• Governmental • Privately owned	Tool of trade
CIIP	Actor responsible of critical national system or infrastructure	Mediocre, Poor	• Governmental • Privately owned	Business risk
Criminal	Computer criminals exploit vulnerabilities for their own, typically financial, gain	Good, Poor	• Individual	Attack method
Researcher	Security researchers actively discover vulnerabilities	Mediocre, Poor	• Individual	Tool of trade
End user	Users are affected by vulnerabilities, and may stumble upon them	Poor	• Individual	Business risk
Security vendor	Vendor of security products: antivirus, IDS/IPS, fuzzer, debugger, exploitation toolkit etc.	Good, Mediocre	• Privately owned	Tool of trade
End user organisation	End user organisations usually discover vulnerabilities as a byproduct of other activities	Good, Poor	• Privately owned	Business risk
Penetration tester	Penetration testers aim to demonstrate weaknesses in systems, often by discovering vulnerabilities in them	Good, Poor	• Privately owned	Tool of trade
Security services provider	Vendor of security services: auditing, compliance, MSS etc	Good, Poor	• Privately owned	Tool of trade
Vendor	Vendor of products or systems	Good, Poor	• Privately owned	Tool of trade
Common good	CERT or similar governmental or independent security organisation	Mediocre, Poor	• Privately owned	Threat

The field

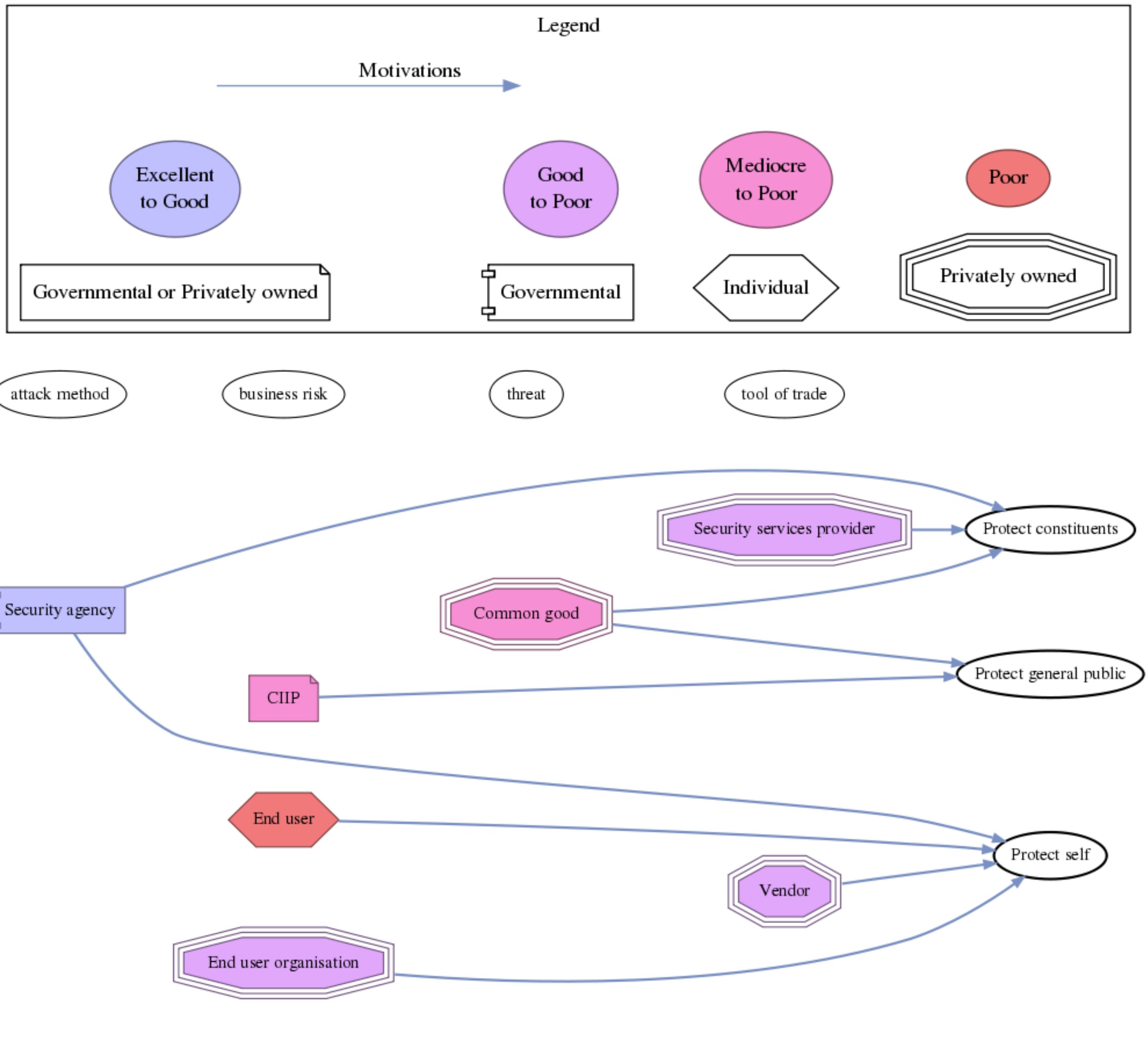


Three quite clear blocks of actors emerge

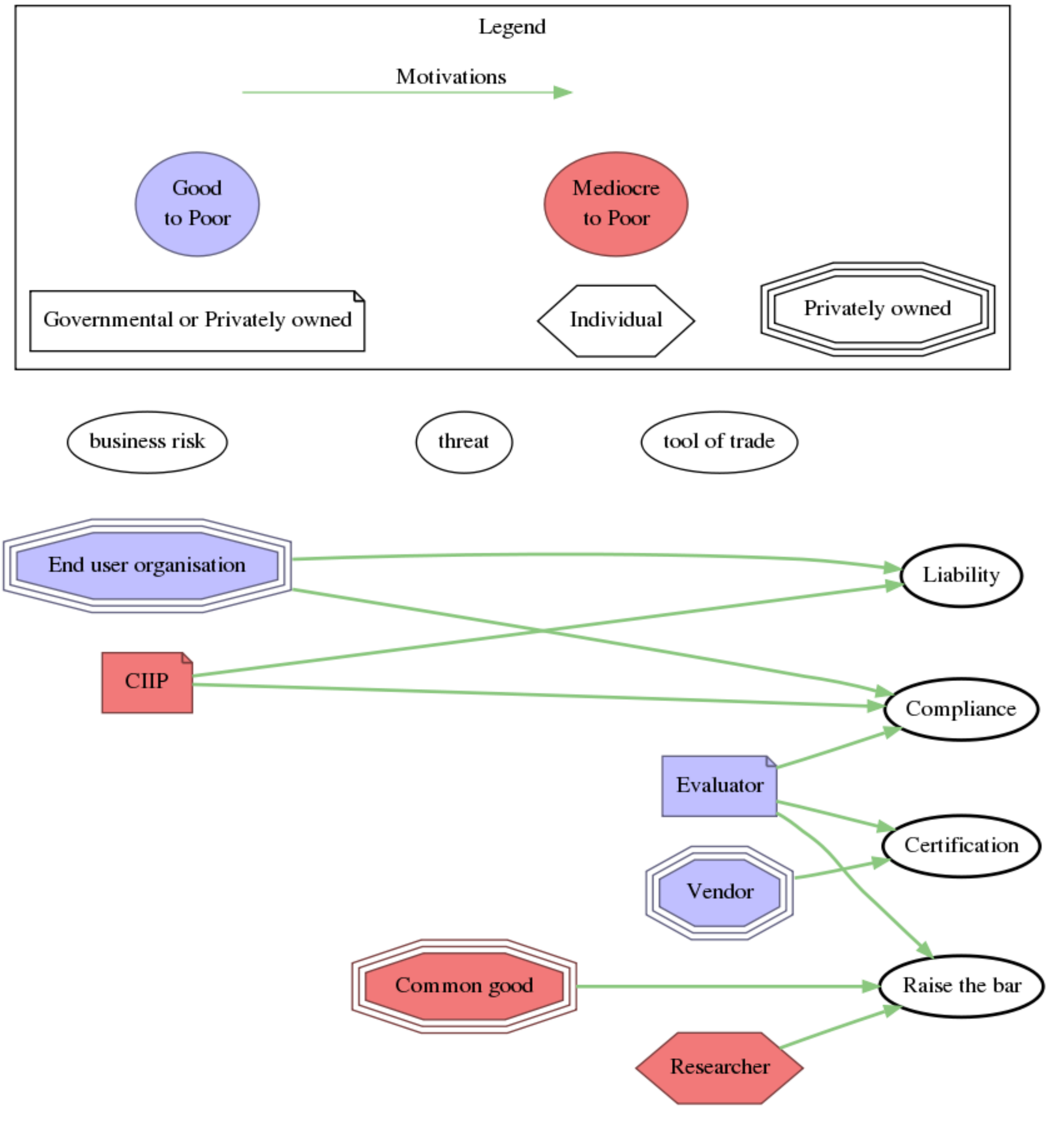
The "offensive block"



The "defensive block"



The "corrective block"



History, how did we end up here

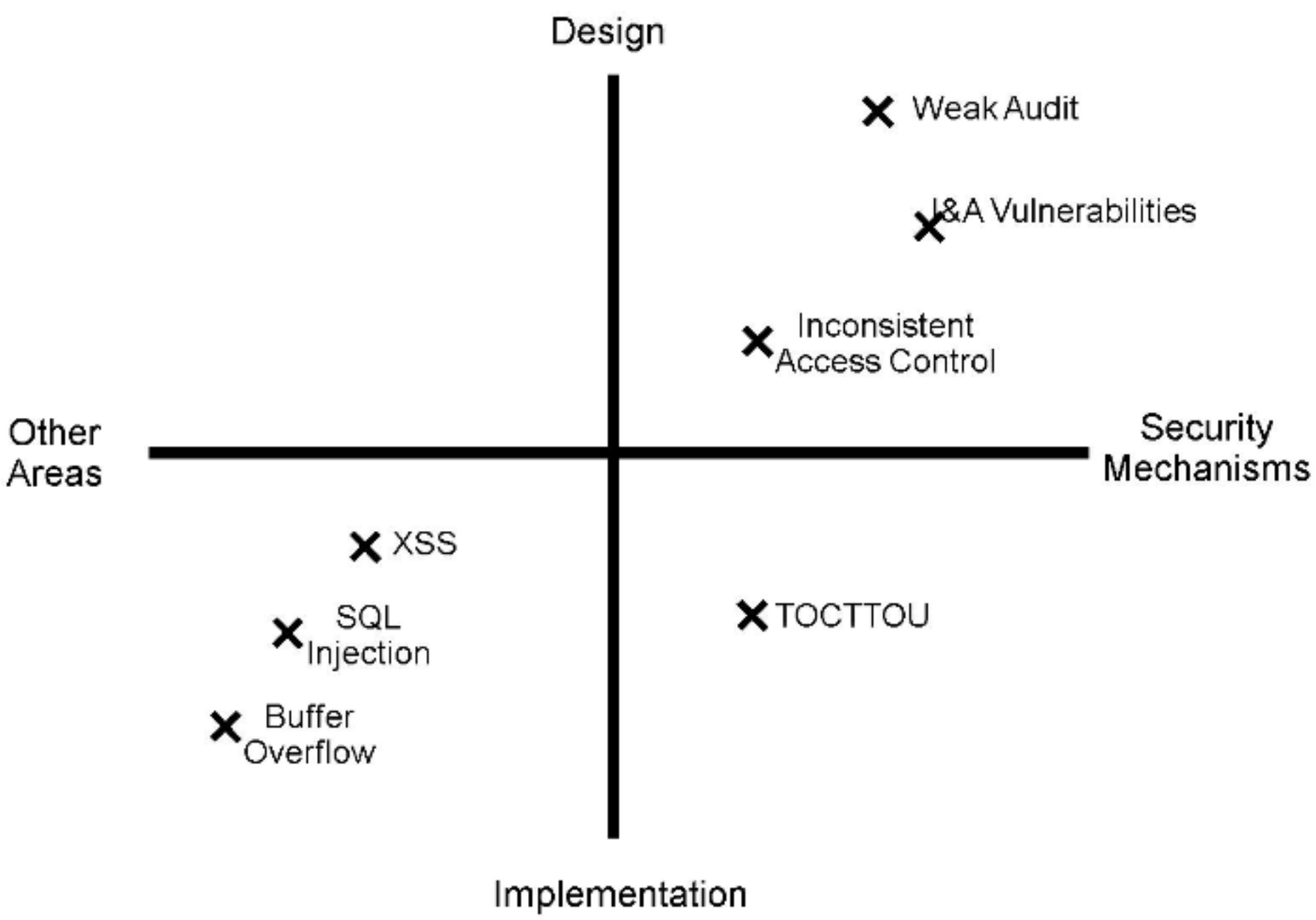
Date	Method	Current state	Resources (avg)
~850	Cryptanalysis	Ever relevant	Medium to Extreme
1952 (or 1842)	Debugging	Ever relevant	Low to High
1953	TEMPEST	Still relevant in some narrow fields (also van Eck/Kuhn/EFPL/LED/Laser/...)	Extreme
1957 (or 1949)	Testing	Ever relevant	Low to High
1972	Fault Injection	Marginal	Medium
1972	Covert Channel	Relevant in crypto applications	Medium to High
1976	Design Review	SDLC, CC, various maturity models	Medium to High
1976	Code Review	SDLC, XP, tool-assisted	Medium
1979	Destructive Testing	Ever relevant	Low to High
1979	Source code analysis tools	Promoted by recent frameworks	Medium to High
1983	Trusted Computer System Evaluation Criteria (Orange book)	ITSEF (1990), CC (2005)	High
1988	Fuzzing	What was this workshop about again?	Low to Medium
1989	Maturity models (SEI-CMM)	Ideas adopted to other models (CLASP, BSIMM, ...)	Medium to High
1999	XP	Lives in Agile methods	Low to Medium
2002	Compiler security extensions	One of the most important security features	Low
2004	SDL	Heavily pushed by MS	Medium to High
2005	Whitebox fuzzing	Promising results (MS, .EXE, 0-knowledge, ...)	Medium to High

Attack simulation

Military	Computer science (trad)	Infosec
Red/tiger team	Hacker	Penetration tester

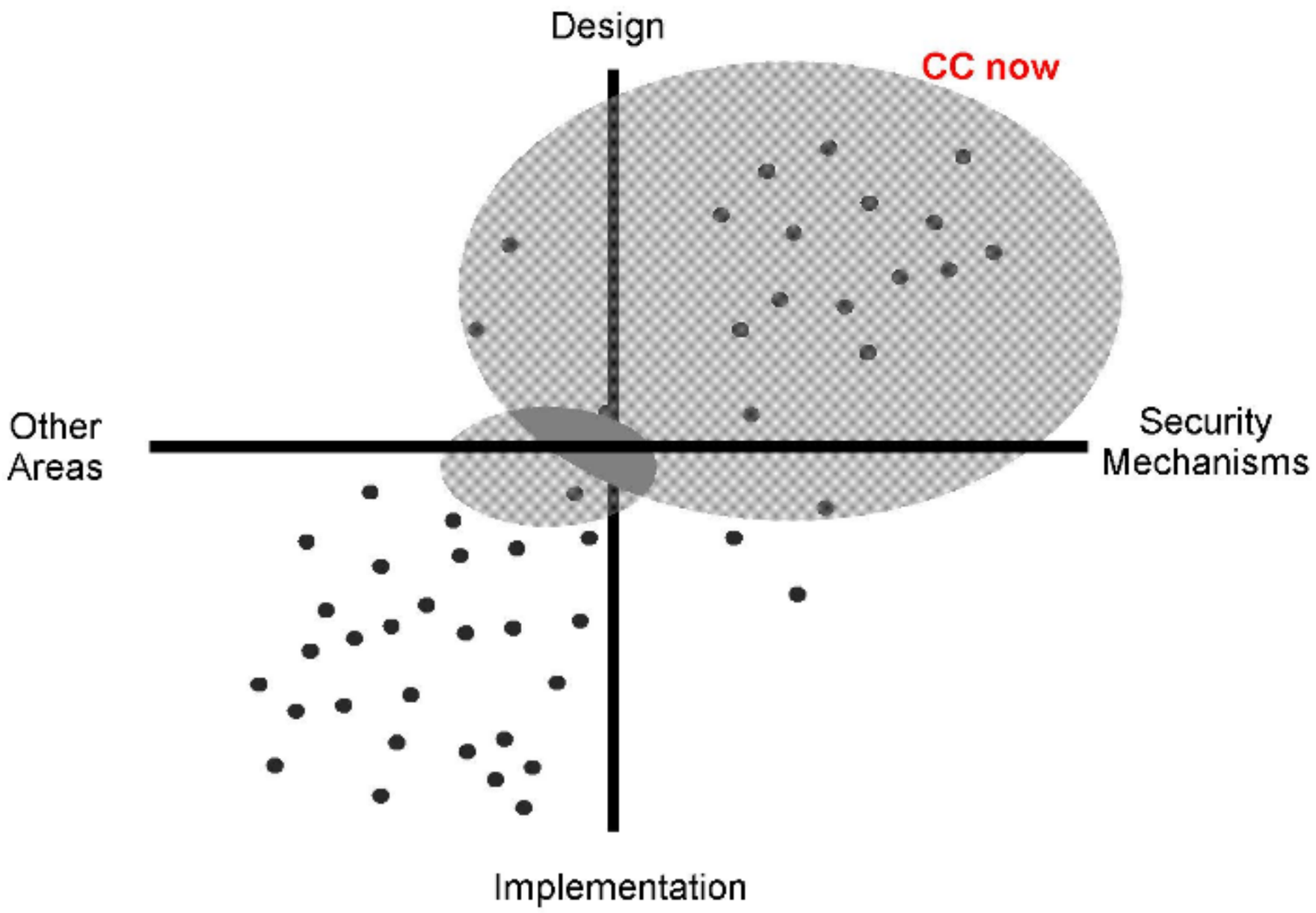
Has good sides, but: a systematic and more comprehensive, real penetration test would cost as much as the attack

Common Criteria Applicability



- Adam O'Brien, Oracle (http://www.commoncriteriaportal.org/iccc/9iccc/pdf/A2412.pdf)

Common Criteria Critique



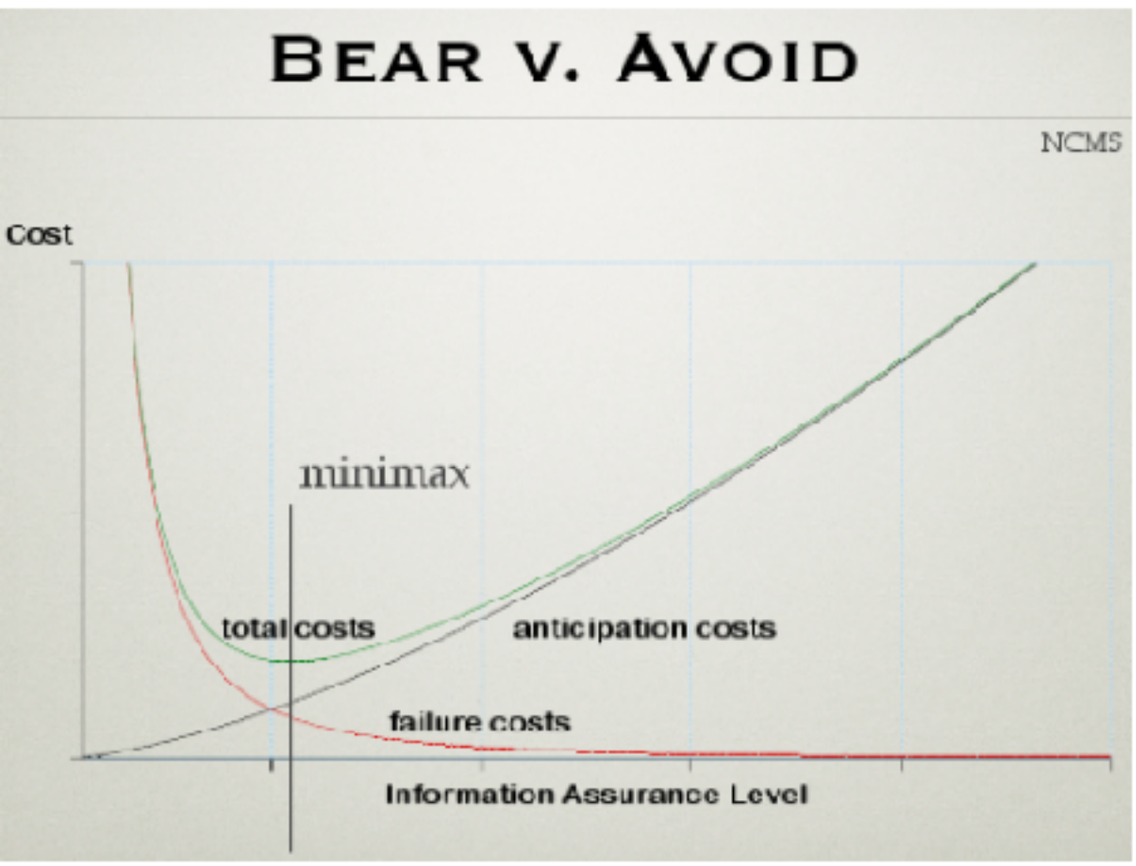
- Adam O'Brien, Oracle (http://www.commoncriteriaportal.org/iccc/9iccc/pdf/A2412.pdf)

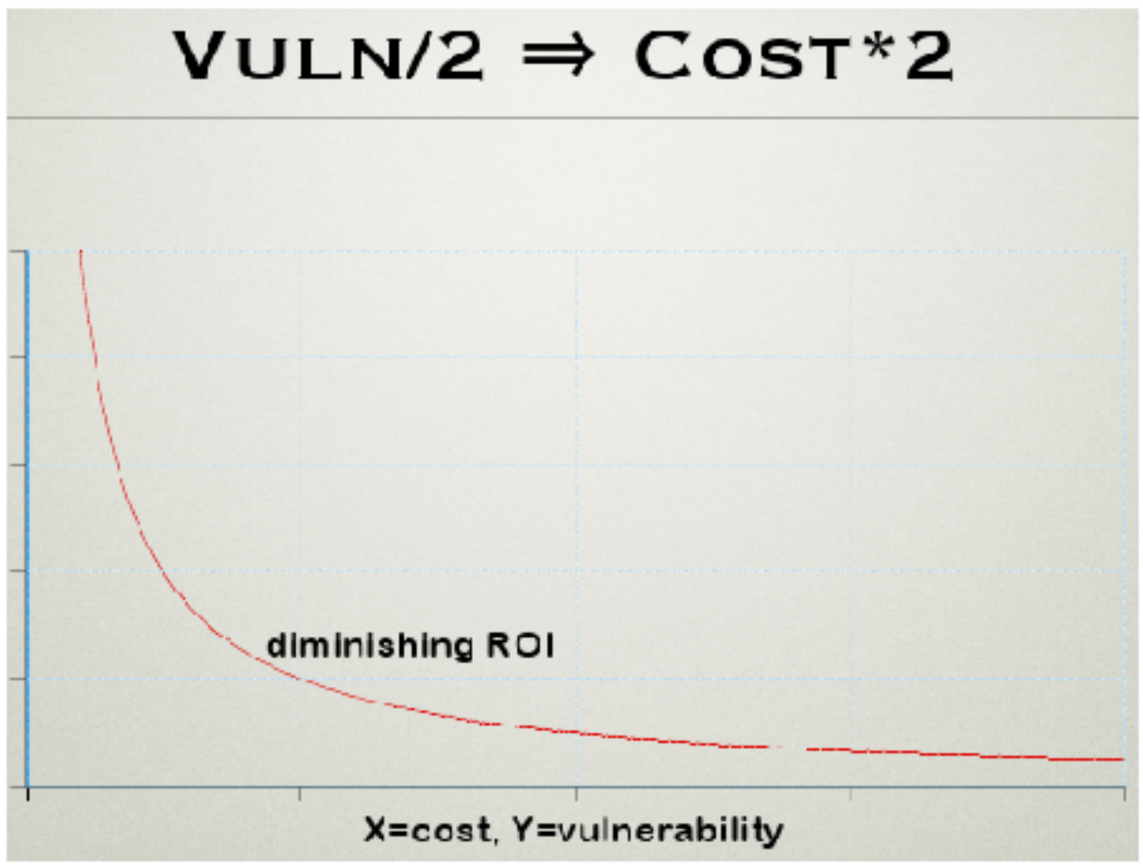
Vulnerability Discovery Methods

- Different methods find different things with different resources (skillz vs. tools)
  - Reading a man page (to get roots)
  - CC EAL5 (electronic microscope)
  - PS3/XBOX etc. DRM hacks
  - Reflection to cost
- "No single VV&T technique can guarantee correct, error-free software. However, a carefully chosen set of techniques for a specific project can help to ensure the development and maintenance of quality software for that project."
  - FIPS 101 (1983)
- "First law: The pesticide paradox. Every method you use to prevent or find bugs leaves a residue of subtler bugs against which those methods are ineffective. "
  - Boris Beizer: Software Testing Techniques (1983)
- A method will first find the low hanging fruit in software, then the returns diminish
  - Once trivial vulnerabilities are eliminated, new classes emerge
- New application types and areas will repeat most errors - learning from the historic vulnerabilities seems to be rare
  - Eg. Buffer overflows mostly gone from MS products, but abundant in ActiveX, embedded, SCADA etc

Resources

- Information security is asymmetrical is like nuclear warhead asymmetrical - a missile is easy to launch, difficult to defend against.
  - But in the end, the one using the most money wins. Using no money is not an option.





- Dan Geer, http://geer.tinho.net/measuringsecurity.tutorial.pdf
- As resources are a premium, discovery method ROI matters
- The ROI of using a single method will decrease over time.
  - This is why discovery portfolios work like defence-in-depth
- Need to be able to measure to justify expenses

Breaking software is like breaking crypto

Not that breaking software is rocket science

Crypto	Vulnerabilities
cryptanalysis	vulnerability analysis
bruteforce	bruteforce

- Though there are intelligent analysis, most vulnerability discover is bruteforce
  - 15min of fame remains an attractive goal
- Efficiency of analysis breaking security is hard to estimate
- Crypto: Breaking SHA hashes with $10000000 would take me x years
  - Measure of the security of tested cryptosystem
- Vulnerabilities: boxen fuzzing protocols in the basement, 10000e development cost, 10000e electricity, ventilation, rent per year
  - Measure of the security of tested software

Asymmetry, Diversification, Refinement

- "Second law: The complexity barrier. Software complexity (and therefore that of bugs) grows to the limits of our ability to manage that complexity. "
  - Boris Beizer: Software Testing Techniques
- Hot markets produce bad code
- Many attackers are using vulnerability discovery automation to the max, while many vendors are still waking up to it
- Finding flaws has become more and more automated, while fixing the issues has become increasingly more difficult due to mounting complexity
- This alone will keep the field going for some time

Choosing tests

- If false negative is serious,
  Then favor sensitivity (& treat false pos)
- If false positive is serious,
  Then favor specificity (& lose false neg)

Multi-stage testing

- Maximizes cost-effectiveness

Stage 1 "screen": dirt cheap, high sensitivity

Stage 2 "confirm": expensive, high specificity

- Combination has higher specificity at expense of sensitivity, *e.g.*, policing the blood supply for HIV

- Dan Geer, http://geer.tinho.net/measuringsecurity.tutorial.pdf

Crypto	Vulnerabilities
SIGINT traffic analysis (S1) to sieve which crypto is worth breaking (S2)	Fuzz for a lot of crashes (S1) to focus exploit development efforts on the most promising ones (S2)?
	Code/spec/design analysis (S1) to focus testing efforts (S2)?

Vulnerability markets

- Everyone can have their cut
  - Develop from crashes
  - Polish and package exploits
    - QA for exploits is 20x more difficult than normal QA - You're doing something you're not supposed in a hostile environment you don't really know of
  - Subscription charge: pay for not disclosing the vulnerabilities
- Reasearchers in developing countries is cheap

Conclusions

Actors and Motivations

In all the identified blocks, actors of the private sector trump governmental/individual actors

(Notable exception: military-industry complex and crime)

- Motivations from the private sector prevail
- Most major innovations stem from the private sector
- Competition drives technological improvement to similar areas

Any Room for Synthesis in the Future?

Common Criteria Moves Towards Fuzzing

- Large number of evaluation requests ->
  - No resources for throughly evaluation ->
    - Ad hoc low assurance level schemes emerge

Interface Analysis + Prioritization + Blackbox Tests

- Common Criteria v4 might include evidence based approach

Code and Design Reviews Rekindled by SDLC and XP

Traditional threats (from-factory trojan, supply chain, side channels, ...) persist

Technical Coverage, Depth vs. Bredth and SocioEconomic Coverage

- Coverage
- NoTechnicalSolutions
- How to cover as many organisations, as many segments, verticals, horizontals, age groups, cultures ...

Testing Coverage

- Input (infinite)
- Code (wildly inaccurate)
- Block (somewhat inaccurate)
- Path
  - Combinatoric explosion in state (variables, memory, ...) and paths
- Spec
  - Compound failures result in combinatoric explosion
- State coverage in dynamic protocols etc.
- **Nobody has actually proven a clear correlation between coverage and the amount of found bugs**
- Testing as a resource question: depth-first might be sexier but breadth-first helps in getting better overall results
  - Anomaly coverage

No Tehnical Measure Will Solve The Problem

- Technical means to tackle the problem
  - ~1 problem
  - ~10 programming/software engineering paradigms (impact in 100 years, needs least resources)
  - ~100 programming languages
  - ~1 000 fundamental ways to shoot yourself in foot
  - ~10 000 popular SDKs/libraries
  - ~100 000 flawed programming books, examples and guidelines
  - ~1 000 000 systems/programs under development
  - ~10 000 000 programmers (Asia!)
  - ~100 000 000 legacy system (impact almost now, needs infinite resources)
- Architecture & Requirements
  - *Security as a new box or bubble* problem -> complexity++
- **Awareness/economics/liability/incentives** ("low" hanging fruit)
- (Testing/Validation/Source Code Analysis)
  - **roughly 50 percent of security problems are the result of design flaws** - Gary McGraw (http://www.cigital.com/papers/download/bsi3-risk.pdf)

Appendix 1 – Roadmap vision and lines of development – tabular form

Vision: **The development and procurement of software and systems which are resilient and sustainable by design, where requirements such as security and privacy are, as a matter of course, defined at project initiation and implemented and assured throughout in risk-based, whole-life processes.**					
Line of development	Activities				
1. Environmental shaping	1.1 Define cost-effective business models.	1.2 Establish procurement strategy, procedures and requirements.	1.3 Manage supply chain risk.	1.4 Establish legislative and regulatory framework.	1.5 Nurture consumer demand.
2. Information exchange and concept development	2.1 Establish mechanisms for information exchange.	2.2 Develop a dynamic library of threats, vulnerabilities, attack patterns and risk models.	2.3 Establish semantics for 'non-functional' requirements engineering.	2.4 Determine whole-life development processes.	2.5 Determine measurable assurance and validation approaches.
3. Technical facilitation	3.1 Utilise secure coding languages	3.2 Develop modelling and analytical tools for planning and assessment.	3.3 Establish trusted libraries of 'reusable code' and components.	3.4 Define interoperability standards for functionality and testing.	3.5 Develop analysis and testing tools for deployed systems and systems of systems.
4. Professionalisation	4.1 Establish the role of 'independent architect'	4.2 Develop national and international standards	4.3 Design curricula for universities and colleges	4.4 Update engineering accreditation core competencies.	4.5 Ensure professional bodies nurture good practice.
5. Communications strategy	5.1 Determine desired behaviours and attitudes of audiences	5.2 Select and analyse audiences	5.3 Determine the message	5.4 Establish interactive communication channels	5.5 Monitor and evaluate communications strategy

-- Cyber Security Knowledge Transfer Network meeting, March 2009

http://www.ktn.qinetiq-tim.net/content/files/events/2009-04-23_building-in-security-assurance-privacy.pdf

Thank you for listening