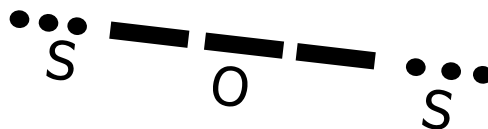
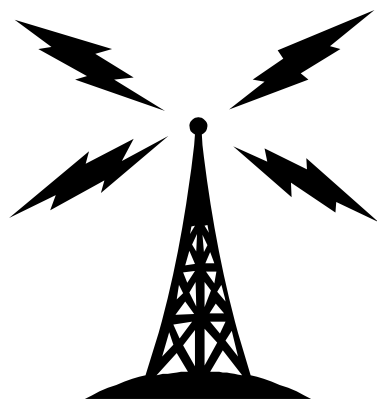


Beyond Hacking: An SOS!

Fred B. Schneider

Samuel B Eckert Professor of Computer Science

Department of Computer Science
Cornell University
Ithaca, New York 14853
U.S.A.



Learning from History: Crisis!

A long time ago ... far, far away...

- Building systems was an art;
- Correctness was the goal.

Learning from History: Resolution!

A long time ago ... far, far away...

- Building systems was an art;
- Correctness was the goal.

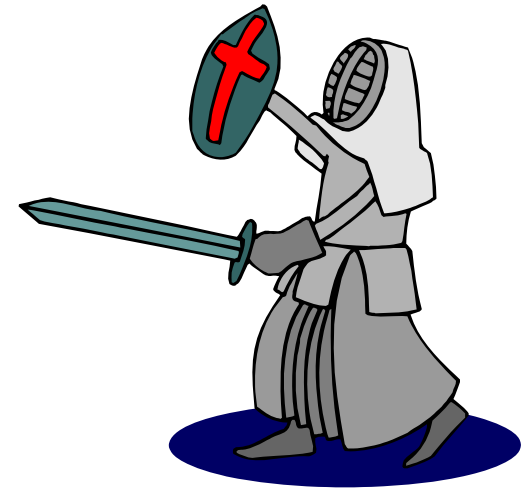
42+ years later "software engineering" offers...

- Nuanced view of the correctness goal
 - Partial and total correctness
 - Safety and Liveness properties
- Quest for engineering produced a **science base**
 - Programming language semantics
 - Model checking and theorem proving
 - Program analysis algorithms
 - ...

History Repeats: Crisis!

*Increasingly we depend on computing systems...
These systems are **not** secure...*

- Defenders react to known attacks.
 - New attack succeeds; we deploy a defense.
... Need to shift from **reactive** to **proactive** mode!
- Effectiveness of defenses cannot be measured.
 - Difficult to justify investments in defenses
 - Difficult to make engineering trade-offs
- Technology base is a moving target.
 - Computers replaced every 3-5 years
 - New deployment environments
 - SCADA, electronic health, ...
 - ... Need insights that transcend technology and applications



Foundations Do Exist!

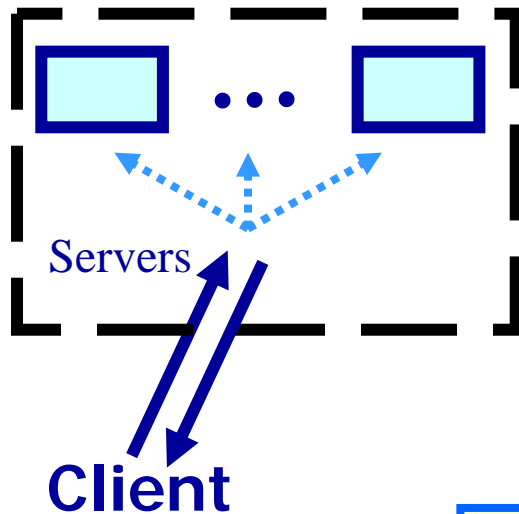
Why can't we just build on...

- Byzantine-fault-tolerance?
- Programming methodology?
- All those years of secure systems building?



Byzantine Fault-Tolerance

Byzantine failure: Arbitrary and malicious behavior, including collusion.



Basic recipe (=implicit assumptions):

- ...
- **Replicas fail independently**
- $2t+1$ replicas tolerate t Byzantine

- *Useful for integrity (access control).*
- *Useless for confidentiality.*
- ***Need: Calculus for independence.***

Program Refinement

If: Pgm sat S ***and*** Pgm' \subseteq Pgm

Then: Pgm' sat S

... depends on (=implicit assumptions!)

- Modeling execution by sequences (or equiv)
- Equating properties (and pgms) with sets of seqs

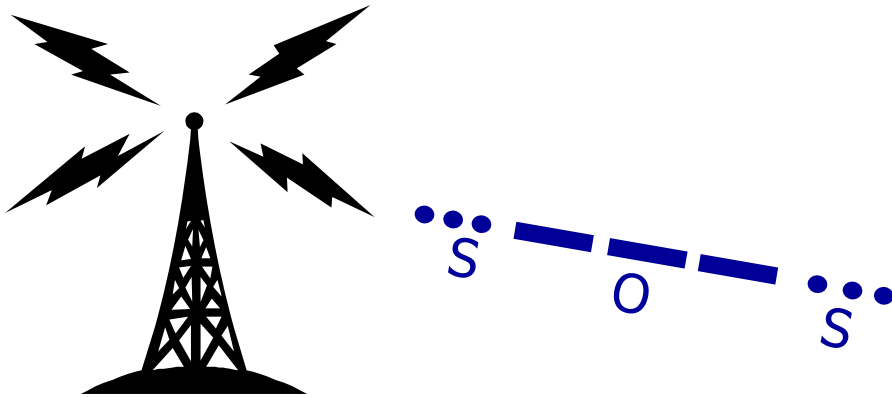
- *Useful for integrity (access control).*
- *Useless for confidentiality.*
- *Need richer model than sets of sequences.*

Our SOS? Evolve the discipline!

Art: Innate abilities and singular talents

Craft: Teachable, due to:

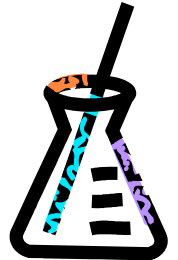
- standardized terminology
- proven techniques



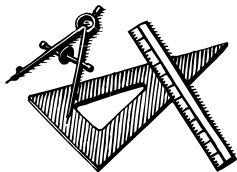
Our SOS? Science Of Security

Science:

- An organized body of knowledge gained through research **-versus-**
- System of acquiring knowledge based on the scientific method **-versus-**
- Laws or theories that are predictive.



Engineering: Craft informed by Science.



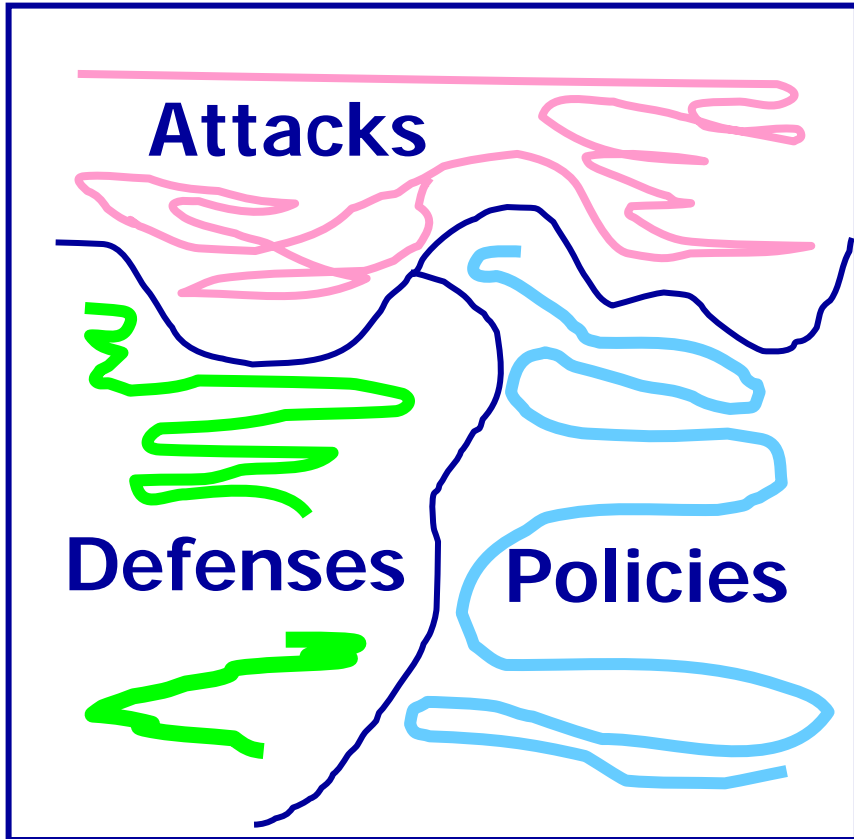
Science Of Security

A **body of laws** that are predictive...

- Transcend specific systems, attacks, and defenses
- Applicable in real settings.
- Provide explanatory value.
 - Abstractions and models
 - Connections and relationships
- Not necessarily quantitative (just like CS)
 - Channel leaks b bits/sec
 - Cannot enforce policy P with mechanism M



Laws About What?



Features:

- Classes of policies
- Classes of attacks
- Classes of defenses

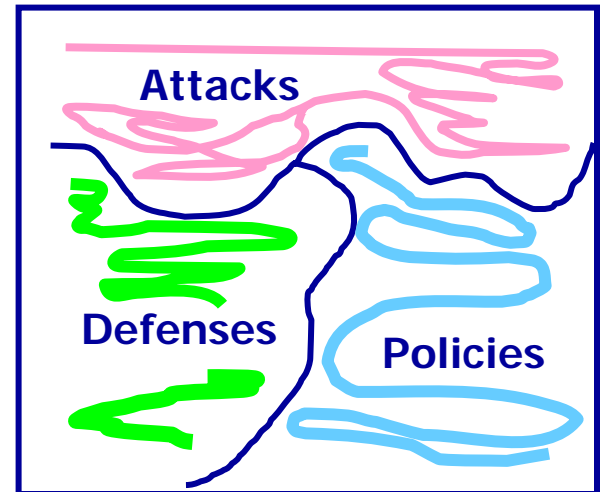
Relationships:

"Defense class D enforces policy class P despite attacks from class A."

"Defense D + Defense D' = ..."

Tour the Landscape

- Limits to Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy
- Foundations of Policy: Policy \leftrightarrow Policy
- Power of Obfuscation: Attack \leftrightarrow Defense \leftrightarrow Policy
- Quantifying Integrity: Policy \leftrightarrow Policy



Monitoring: Attack ↔ Defense ↔ Policy

Execution Monitoring (EM)

[Schneider 2000]

Execution monitor:

- Gets control on every policy-relevant event
- Blocks execution if allowing event would violate policy
- Integrity of EM protected from subversion.

Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy

Classical View of Properties

System behavior t : an infinite trace

$$t = s_0 s_1 s_2 s_3 \dots s_i \dots$$

System property P : set of traces

$$P = \{ t \mid \text{pred}(t) \}$$

System S : set S of traces (its behaviors).

System S **satisfies** property P : $S \subseteq P$

Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy

Safety and Liveness

[Lamport 77]

Safety: Some “bad thing” doesn’t happen.

Liveness: Some “good thing” does happen.

Safety and Liveness

[Alpern+Schneider 85,87]

Safety: Some “bad thing” doesn’t happen.

- Proscribes traces that contain some irremediable prefix.

Liveness: Some “good thing” does happen.

- Prescribes that prefixes are not irremediable.

Thm: Every property is the conjunction of a safety property and a liveness property.

Thm: Safety properties proved by invariance.

Thm: Liveness properties proved by well-foundedness.

Monitoring: Attack ↔ Defense ↔ Policy

Execution Monitoring (EM)

[Schneider 2000]

Execution monitor:

- Gets control on every policy-relevant event
- Blocks execution if allowing event would violate policy
- Integrity of EM protected from subversion.

Monitoring: Attack ↔ Defense ↔ Policy

Execution Monitoring (EM)

[Schneider 2000]

Execution monitor:

- Gets control on every policy-relevant event
- Blocks execution if allowing event would violate policy
- Integrity of EM protected from subversion.

Conclusion:

- Acceptance based **solely** on the current execution
- Rejection based on **solely** prefix of execution

Thm: EM only enforces safety properties.

Monitoring: Attack ↔ Defense ↔ Policy

Execution Monitoring (EM)

[Schneider 2000]

Execution monitor:

- Gets control on every policy-relevant event
- Blocks execution if allowing event would violate policy
- Integrity of EM protected from subversion.

Examples of EM-enforceable policies:

- Only Alice can read file F.
- Don't send msg after reading file F.
- Requests processing is FIFO wrt arrival.

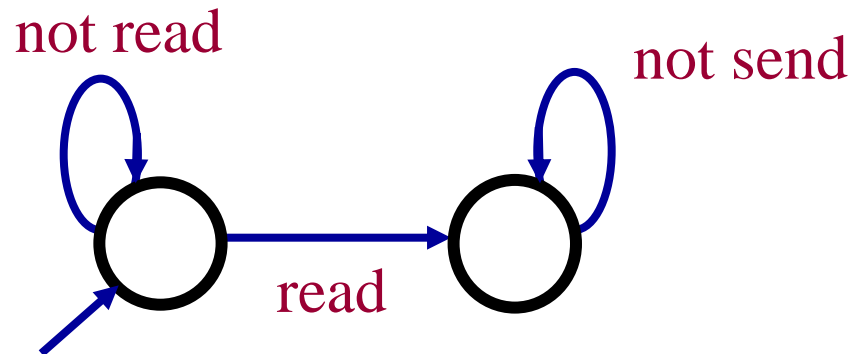
Examples of non EM-enforceable policies:

- Every request is serviced
- Value of x is not correlated with value of y.
- Avg execution time is 3 sec.

Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy

New EM Approaches

Every safety property corresponds to an automaton.



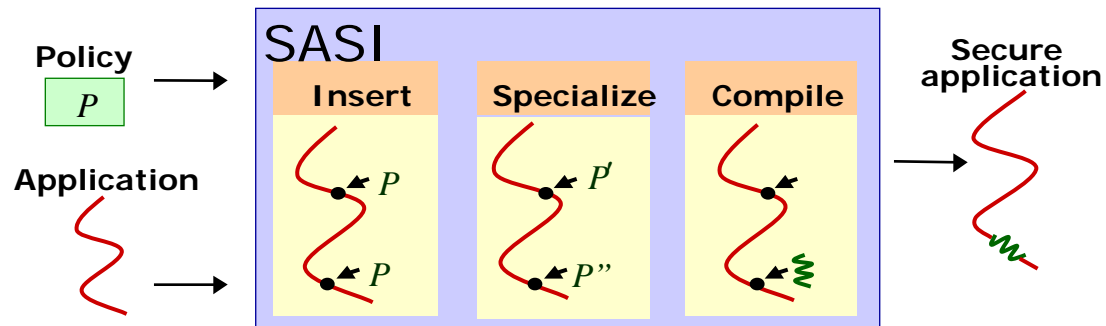
$\square(\text{read} \Rightarrow \square \neg \text{send})$

Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy

Science \rightarrow Engineering

New approach to enforcing EM policies:

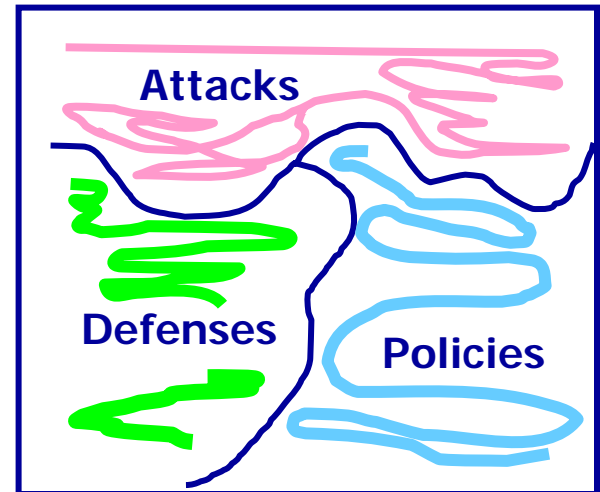
1. Automaton \rightarrow Pgm code (case statement)
2. Inline automaton into target program.



Relocates trust from pgm to reference monitor.

Tour the Landscape

- Limits to Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy
- Foundations of Policy: Policy \leftrightarrow Policy
- Power of Obfuscation: Attack \leftrightarrow Defense \leftrightarrow Policy
- Quantifying Integrity: Policy \leftrightarrow Policy



Security Features: Policies

Policy: What the system should do; what the system should not do:

- **Confidentiality:** Who is allowed to learn what?
- **Integrity:** What changes are allowed by system.
... includes resource utilization, input/output to environment.
- **Availability:** When must service be rendered.

Usual notions of "program correctness" are a special case.

Security \neq Safety Properties

Non-correlation: Value of L reveals nothing about value of H.

Non-interference: Deleting cmds from H-users cannot be detected by cmd exec by L-users.

[Goguen-Meseguer 82]

Properties, safety, liveness not expressive enough!

EM not powerful enough.

Hyper-Properties

[Clarkson+Schneider 08]

Hyper-property: set of properties
= set of sets of traces

System S satisfies hyper-property HP: $S \in \text{HP}$

Hyper-property $[P]$: $\{P' \mid P' \subseteq P\}$

Note:

- $(P \in \text{HP} \text{ and } P' \subseteq P) \Rightarrow \text{HP}$ not required.
- Non-interference is a HP.
- Non-correlation is a HP.

Hyper-Safety Properties

Hyper-safety HS: “Bad thing” is property M comprising finite number of finite traces.

- Proscribes tracing containing irremediable observations.

Thm: For safety property S , $[S]$ is hyper-safety.

Thm: Not all hyper-safety are refinement closed.

Hyper-Safety Applications

2SP: Safety property on program S composed with itself
(with variables renamed). [Terauchi+Aiken 05]

$S; S'$

2SP transforms information flow into a safety property!

K-safety: Safety property on program

$S^K: S \parallel S' \parallel \dots \parallel S''$

K-safety is HS.

Thm: Any K-safety property of S is equivalent to a safety property on S^K .

Hyper-Liveness Properties

Hyper-liveness HL: Any finite set M of finite traces has an augmentation that is in HL.

Prescribes: observations are not irremediable.

- Examples: possibility, statistical performance, etc.

Thm: Every HP is the conjunction of HS and HL.

Hyper Properties Questions

Q: Verification for HS and HL?

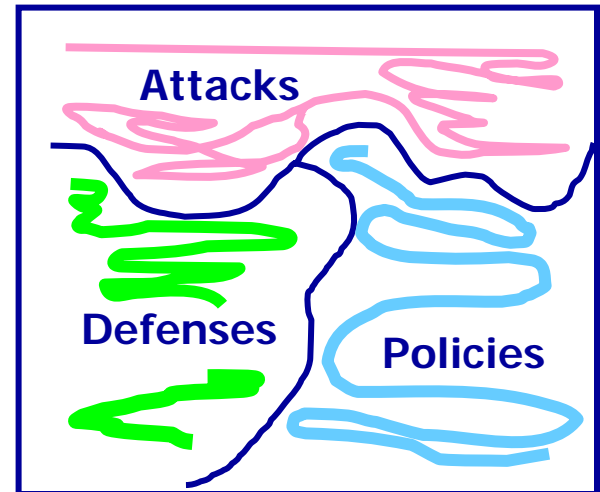
Q: Refinement for HS and HL?

Q: Enforcement for HS and HL?



Tour the Landscape

- Limits to Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy
- Foundations of Policy: Policy \leftrightarrow Policy
- Power of Obfuscation: Attack \leftrightarrow Defense \leftrightarrow Policy
- Quantifying Integrity: Policy \leftrightarrow Policy



Obfuscation: Attack ↔ Defense ↔ Policy

Obfuscation: Goals and Options

Semantics-preserving random program rewriting...

Goals: Attacker does not know:

- address of specific instruction subsequences.
- address or representation scheme for variables.
- name or service entry point for any system service.

Options:

- Obfuscate source (arglist, stack layout, ...).
- Obfuscate object or binary (syscall meanings, basic block and variable positions, relative offsets, ...).
- All of the above.

Obfuscation Landscape

[Pucella+Schneider 06]

Given program S , obfuscator computes **morphs**:
 $T(S, K_1), T(S, K_2), \dots T(S, K_n)$

- **Attacker knows:**
 - Obfuscator T
 - Input program S
- **Attacker does not know:**
 - Random keys $K_1, K_2, \dots K_n$
... Knowledge of the K_i would enable attackers to automate attacks!

Will an attack succeed against a morph?

- Seg fault likely if attack doesn't succeed.
integrity compromise → availability compromise.

Successful Attacks on Morphs

All morphs implement the same interface.

- **Interface attacks.** Obfuscation cannot blunt attacks that exploit the semantics of that (flawed) interface.
- **Implementation attacks.** Obfuscation can blunt attacks that exploit implementation details.

Def. implementation attack: An input for which all morphs (in some given set) don't **all** produce the same output.

Obfuscation: Attack ↔ Defense ↔ Policy

Effectiveness of Obfuscation

Ultimate Goal: Determine the probability that an attack will succeed against a morph?

Modest goal: Understand how effective obfuscation is as compared with other defenses?

- Obvious candidate: Type checking

Type Checking as a Defense

Type checking: Process to establish that all executions satisfy certain properties.

- Static: Checks made prior to exec.
 - Requires a decision procedure
- Dynamic: Checks made as exec proceeds.
 - Requires adding checks. Exec aborted if violated.

Probabilistic dynamic type checking: Some checks are skipped on a random basis.

Obfuscation: Attack ↔ Defense ↔ Policy

Obfuscation versus Type Checking

Thesis: Obfuscation and probabilistic dynamic type systems can “defend against” the same attacks.

From “thesis” → “theorem” requires fixing:

- a language
- a type system
- a set of attacks

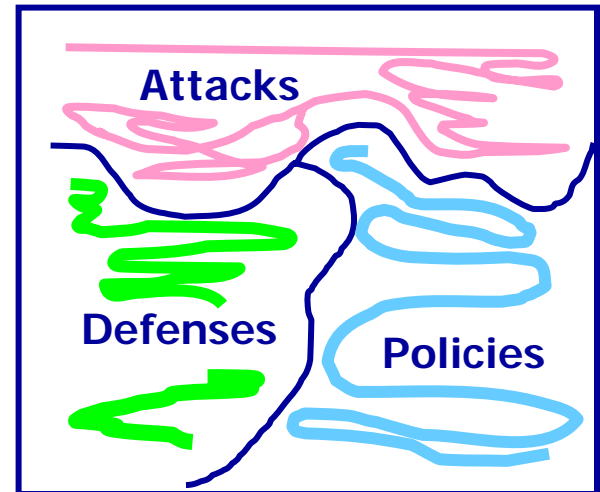
Obfuscation: Attack ↔ Defense ↔ Policy

Pros and Cons of Obfuscation

- Type systems:
 - Prevent attacks (always---not just probably)
 - If static, they add no run-time cost
 - Not always part of the language.
- Obfuscation
 - Works on legacy code.
 - Doesn't always defend.

Tour the Landscape

- Limits to Monitoring: Attack \leftrightarrow Defense \leftrightarrow Policy
- Foundations of Policy: Policy \leftrightarrow Policy
- Power of Obfuscation: Attack \leftrightarrow Defense \leftrightarrow Policy
- Quantifying Integrity: Policy \leftrightarrow Policy



Qualitative vs Quantitative:

Quantifying Integrity

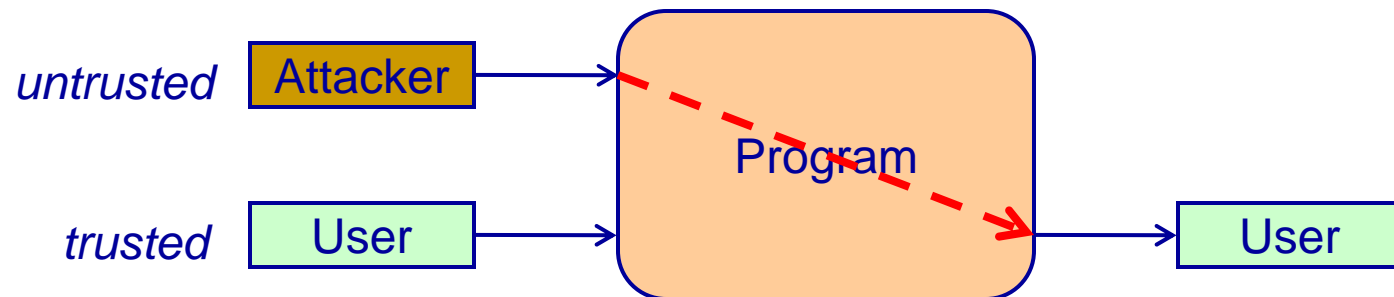
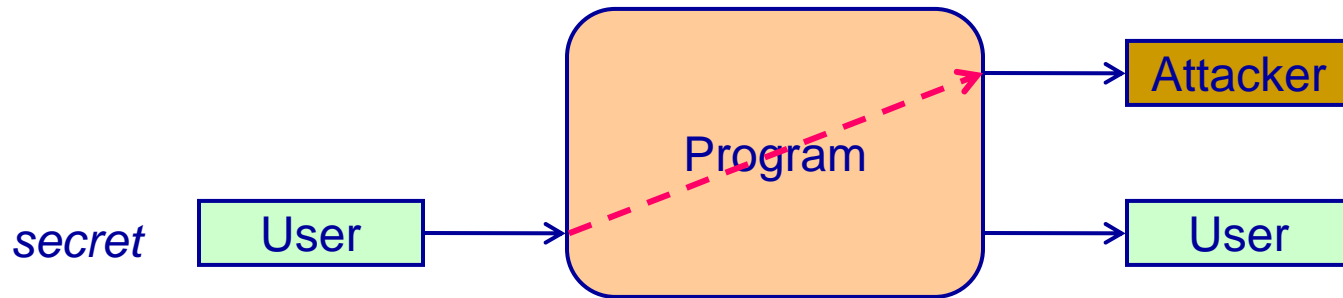
“When you can **measure what you are speaking** about, and express it in numbers, **you know something about it**; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre [sic] and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science.”

William Thomson, 1st Baron Kelvin

From “Electrical Units of Measurement”, a lecture delivered the Institute of Civil Engineers, London, May 3, 1883.

Qualitative vs Quantitative:

Confidentiality and Integrity



Turning the Crank?

*Joint work with Michael Clarkson. To appear 2010 Computer Security Foundations

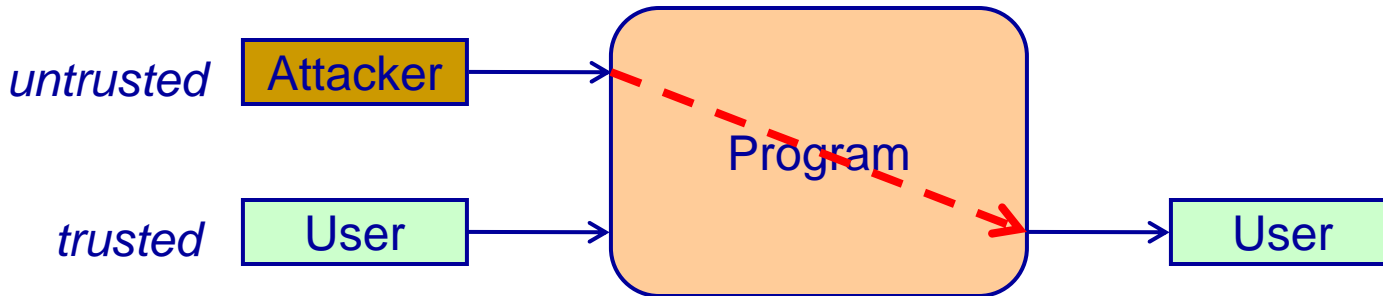
- Confidentiality / Integrity Duality [Biba]
 - Confidentiality: Don't read **secret**, write **public**.
 - Integrity: Don't read **untrusted**, write **trusted**.
- Theories exist for quantifying flows of secret info to public outputs.
 - Based on Shannon information theory
- Apply Biba duality and obtain means to measure flows of **untrusted** info to trusted **outputs**.

Qualitative vs Quantitative: The Duality!

*Joint work with Michael Clarkson. To appear 2010 Computer Security Foundations

Attacker consequences:

- **Contamination** (dual of leakage) ⋯→
 - Output := (t, u)
 - ... *Predict untrusted input from trusted input and trusted output*

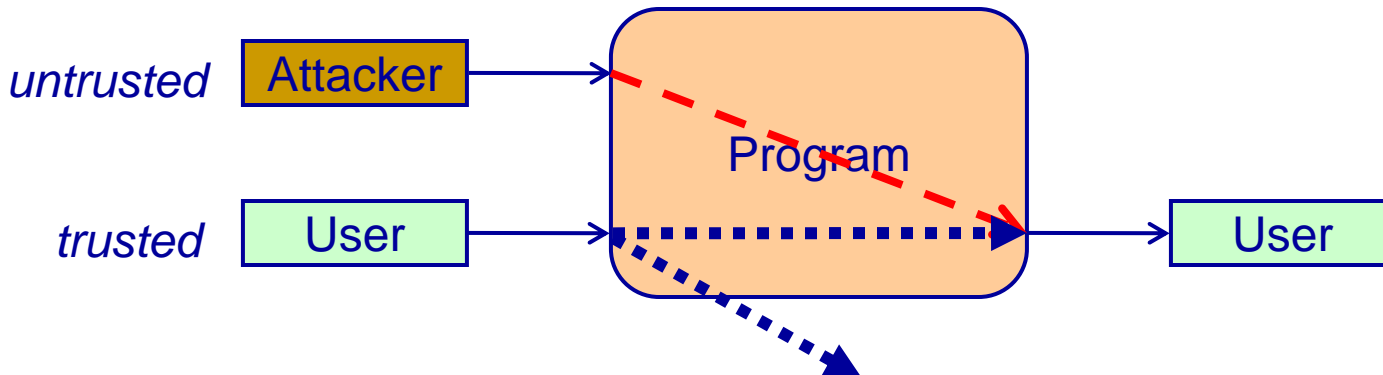


The Duality is incomplete...

*Joint work with Michael Clarkson. To appear 2010 Computer Security Foundations

Attacker consequences:

- **Contamination** (dual of leakage) $\cdots \rightarrow$
 - $\text{Output} := (t, u)$
... Predict untrusted input from trusted input and trusted output
- **Suppression** (trusted input suppressed from trusted output):
 - $n := \text{rand}(); \text{Output} := t \text{ XOR } n$ $\cdots \rightarrow$
... Predict trusted input from trusted output.
- Both contamination and suppression
 - $\text{Output} := t \text{ XOR } u$



Qualitative vs Quantitative:

Law: Leakage vs Suppression

*Joint work with Michael Clarkson. To appear 2010 Computer Security Foundations

Declassifier: program that reveals some information but suppresses the rest.

What isn't leaked is suppressed...

LS Thm: Leakage + Suppression = Constant

Science → Engineering

*Joint work with Michael Clarkson. To appear 2010 Computer Security Foundations

Statistical databases anonymize query results:

- Use suppression to avoid leakage
- Sacrifice integrity for confidentiality's sake.

LS Thm provides basis for eval and comparison.

- **K-anonymity** [Sweeney '02]
 - Doesn't bound leakage or suppression
- **Entropy L-diversity** [Machanavajjhala et al. '07]
 - Suppresses at least L bits of information about individual
- **Differential Privacy** [Dwork '06]
 - ??????

Role of Experimentation?

- In natural sciences:
 - Validate laws wrt unknown reality
- In computer science:
 - Measurements to learn about (complex) realities we have created.
 - Prototyping validates assumptions (esp unstated ones)
- In computer security, add:
 - Attacks provide “data” (inspiration?) for defenders.

Asking the Right Questions

- Prove: Trust cannot be created, it can only be relocated.
 - basis for composing defenses and trust relocation.
- Can sufficiently introspective active defences always be subverted?
 - Consequences for HIV / AIDS / cancer.
- When are components independent?

Reading

- Independence from obfuscation: A semantic framework for diversity. *Proceedings 19th IEEE Computer Security Foundations Workshop* (Venice, Italy), July 2006, 230--241. With Riccardo Pucella.
- Enforceable security policies. *ACM Transactions on Information and System Security* 3, 1 (February 2000), 30--50.
- SASI enforcement of security policies: A retrospective. *Proceedings of the New Security Paradigms Workshop* (Caledon Hills, Ontario, Canada, September 1999), Association for Computing Machinery, 87--95. With Ulfar Erlingsson.
- IRM enforcement of Java stack inspection. *Proceedings 2000 IEEE Symposium on Security and Privacy* (Oakland, California, May 2000), IEEE Computer Society, Los Alamitos, California, 246--255. With Ulfar Erlingsson.
- Hyperproperties. *Proceedings 21st IEEE Computer Security Foundations Symposium* (Pittsburgh, PA, June 2008), 51--65. With Michael Clarkson.
- Quantification of Integrity. *Proceedings Computer Security Foundations Symposium* (Edinburgh, U.K. July 2010. To appear. With Michael Clarkson.