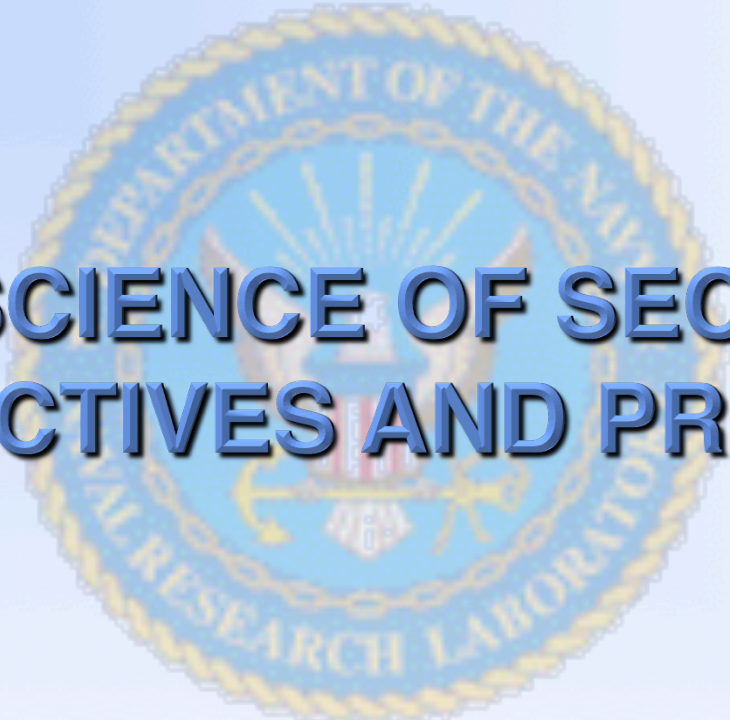




THE SCIENCE OF SECURITY PERSPECTIVES AND PROSPECTS

JOHN D. MCLEAN
Information Technology Division
Naval Research Laboratory

OUTLINE

What would we like to have?

- ◆ What is a science of computer security?

What can history teach us: two case studies from 1987 - 1994

- ◆ Access Control Models
 - A near miss
- ◆ Information Flow Models
 - A success

Are we safe? Have we shown that security a science?

- ◆ Limitations, concerns, and ways forward

What would we like to have?

What is Science?

This Symposium's View: We interpret science in the broadest sense of the systematization of knowledge to uncover foundational principles through theory-driven inquiry.

Wikipedia's View: Science is a systematic enterprise that builds and organizes knowledge in the form of testable explanations and predictions about the universe.

Caveat: The term *computer science* didn't appear in print until 1959 and computer security came much later.

- ◆ Contrast this with physics which had over 2,000 years to progress from Aristotle's *Physics* to Newton's *Principia*, another 175 years after that to develop Maxwell's equations, and an additional 150 years to build those.

What I think we're after

A systematization of knowledge

- ◆ An integrated set of fundamental principles and other principles that are derivable from them. We want a coherent framework rather than a disparate collection of ad hoc parts.

Fundamental principles that are predictive.

- ◆ Note that if they're predictive, they're testable. We want to be able to justify the framework in a repeatable, independent fashion.
- ◆ What makes our framework scientific is not that we have foundational principles, but the fact that those principles can (and are) challenged and tested with objective results.
 - The scientific community both validates and sets standards with respect to what's interesting, what's an explanation, what is acceptable confirmation, etc.

Ultimately, we want fundamental principles that are explanatory

- ◆ It's the difference between the periodic table and the valence theory of electrons or between Mendel and the theory of genetics. We want to understand why things work so we can safely generalize.

Sorts of questions I think we'd like to answer in a reasonably “scientific” way

How secure is my system?

- ◆ What, if any attacks is my system vulnerable to and how difficult would it be to carry out a successful attack?

If I do x, how will it affect the security of my system?

- ◆ how much will it reduce or increase the probability of a successful attack of type y?
- ◆ how much will it reduce or increase the cost of performing a successful attack of type y?
- ◆ What, if any, attacks will I eliminate completely?

How can I build a secure system?

- ◆ If I have a secure specification, how can I implement it securely?
- ◆ If I compose systems x and y, what are the security implications.

What can history teach us?

The case for history

Why look back?

- ◆ What's past is prologue.

Why these two examples and why 1987 - 1994?

- ◆ I happen to have first-hand experience of these examples
- ◆ They unfolded at the IEEE Symposium on Security and Privacy, which was the scientific community for computer security. For whatever reason, they were part of, or at least contemporaneous with a turning point for that community and possibly for the science of computer security.
 - For the 30 year period 1980 – 2009, 14 of the 20 top cited papers from the IEEE Symposium on Security and Privacy are from the eight year period of 1987 – 1994.
 - 25% of the Symposium's top 20 papers for 1980 – 2009 were directly involved in the two examples I'm going to discuss.
 - Specifications for Multi-Level Security and a Hook-Up Property (McCullough 1987)
 - Noninterference and the Composability of Security Properties (McCullough 1988)
 - Security Models and Information Flow (McLean, 1990)
 - A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions (McLean, 1994)
 - Reasoning About Security Models (McLean, 1987)

Historical Case Study 1: Access Control Models*

Access Control 1

Following the paper and safe world, confidentiality and, to some extent, integrity were to be enforced by access control.

- ◆ Access control lists in Multics [Neumann]
- ◆ Capability Lists (MIT PDP-1)
- ◆ Access Control Matrix [Lampson, 1971]
 - subjects (rows in a matrix representing users and programs)
 - objects (columns in a matrix representing subjects and files)
 - access rights
 - transformation rules (rules for altering the matrix)
 - Part of the appeal was that it fit in nicely with the Reference Monitor concept (Anderson, 1972) and the Security Kernel concept (Schell, 1973)

The first theoretical analysis of access control was performed by Harrison, Ruzzo, and Ulman (HRU) in 1976.

- ◆ The general problem of whether an initial configuration can eventually transition to a given arbitrary configuration is undecidable.

Access Control 2

The undecidability of the HRU model led to the proliferation of many models that restricted the Lampson model in some way.

- ◆ Monotonic/Monoconditional HRU [Harrison, Ruzzo, and Ulman]
- ◆ Take-Grant Model [Jones, Lipton, and Snyder]
- ◆ Schematic Protection Model [Sandhu]
- ◆ Typed Access Matrix Model [Sandhu]

The most famous of these limited models was the Bell and LaPadula Model (BLP), which supplemented HRU with a lattice of security levels. In its simplest formulation BLP consisted of:

- ◆ subjects and objects a la HRU
- ◆ 2 access rights (read, write)
- ◆ 2 State Restrictions
 - simple security: a subject cannot have read access to an object above its security level
 - *-property: a subject cannot have write access to an object below its security level

Access Control 3

BLP seemed to capture our intuitive needs with respect to confidentiality

- ◆ Neither people nor programs can read things they are not entitled to read
- ◆ Programs cannot write information to lower levels
 - A Trojan Horse loaded onto a secret system could not write information from a classified file to an unclassified file

The independent verification for BLP was a theorem, called the “Basic Security Theorem” (BST), that if a system starts out in a BLP-secure state and obeys certain restrictions when transitioning from one state to another, it can never reach a non-BLP-secure state.

- ◆ This certainly seems scientific in that we have model (set of principles) and a mathematical theorem that provides objective independent validation of the model.

Access Control 4

BST seemed to justify BLP's state-based formulation security because it seemed to give an independent transformation-based formulation of security that was coextensive with BLP's

- ◆ Similar to justification of recursiveness as an explication of computability because of its coextensiveness with Turing computability and with λ -definability.
- ◆ Similar to justification of first order provable formula as a explication of first order logical truth because of its coextensiveness with valid formula

The trouble with BST as a justification of BLP was that it was really not an alternative explication of security at all. Although it was correct, it was transparent with respect to the definition of a “secure state”.

- ◆ It gave no more justification to BLP than it would give to a model that negated BLP's two restrictions. [McLean 1985]

So, what happened in 1987?

The first truly transformation-based formulation of BLP [McLean]

- ◆ If a state transition introduces new read or write rights, then the new rights can not violate simple security or *-security in the current state and no security levels may be changed by the state transition
- ◆ If a state transition alters subject security levels, the new levels cannot violate simple security or *-security in the current state and the transition can not alter any object's security level or add enter/delete any rights.
- ◆ If a state transition alters object security levels, the new levels cannot violate simple security or *-security in the current state and the transition can not alter any subject's security level or add enter/delete any rights.

This seems like a trivial approximation to BLP, but it's sufficiently different to show a flaw in BLP:

- ◆ System Z: whenever a subject requests access to an object, downgrade all subjects and objects to the lowest possible security level and grant the access
- ◆ System Z satisfies the state-based formulation of BLP but fails to satisfy the transformation-based version.

Why was this important?

System Z was the result of an attempted independent verification of evidence (BST) rendered in support of a theory (BLP).

However, perhaps, the most important take away from this episode was the community reaction.

- ◆ Debate was so intense it generated dedicated panel sessions, mention in the popular press and the security semi-popular press, and even a sociological analysis.
 - This is not indicative of a mature scientific field.

There was a silver lining, however – creation of the annual Foundations of Computer Security Workshop, partly in response to an open letter by Don Good criticizing the community response.

Was this science?

Pro:

- ◆ HRU created a very general framework for access control and proved a fundamental limitation theorem.
- ◆ BLP put forward the BST, which seemed to establish the model's correctness in an independent, verifiable way.
- ◆ The support that BST provided BLP was examined

Con:

- ◆ BST was shown to be an insufficient support and BLP was shown to be flawed.
- ◆ Even if there was disagreement in the community about what to make of system Z, the disagreement itself showed that our fundamental principles were, at least, ambiguous. This is not indicative of a mature science, even if we take into account the fact that the history of physics and mathematics are both messy at times.

Pro:

- ◆ The community convened an annual symposium to look at scientific underpinnings of security.

Epilogue

The BLP episode, in fact, did produce at least one generalized access control model based on BLP that allowed changes to security levels [McLean, 1988], but the model was never pursued for a couple of reasons.

- ◆ Restrictions on changes to security levels are very specific to the application.
 - The community moved to more application-specific security models where rules governing alteration of security levels, privileges, etc could be spelled out in more detail.
- ◆ Covert channels rendered access control models less appealing than they initially seemed to be
 - Subjects can glean information about high-level activity without having direct access to high-level files
 - A Trojan Horse operating at a high security level can pass high level information to low level users without having direct access to any low-level objects
 - Such attacks are also possible in the paper and safe world, but they would be too slow, obvious, and baroque (when compared to alternatives) to worry about

Historical Case Study 2: Information Flow Models*

Information Flow Models 1

Covert channels showed the need to define confidentiality in terms of information flow, rather than in terms of accesses.

- ◆ Basic concept is that within the system information cannot flow from high level users to low level users.
- ◆ Access control models can be seen as safety property approximations of information flow.

The concept of an information flow goes back to Fenton (1974) and Denning (1975), but the first information flow model was Noninterference (Goguen and Meseguer, 1982). Basic elements of the model were:

- ◆ Traces of user inputs
- ◆ An output function that maps a trace to a system output
- ◆ A purge function that removes high level inputs from a trace
- ◆ The requirement that for all traces T , $\text{low-output}(T) = \text{low-output}(\text{purge}(T))$

Information Flow Models 2

For deterministic systems, classical Noninterference is almost perfect for confidentiality

- ◆ With respect to security it's a bit too strong in that it assumes a worse case scenario, but this can usually be finessed
 - Interference enables communication only if interference can be controlled by a sender and detected by a receiver
 - The first information theoretic analysis of Noninterference was presented by Millen at the first Foundations of Computer Security Workshop

However, classical Noninterference's assumption of strict determinism severely restricted its use

- ◆ Useless for most realistic specifications and for systems that use probabilistic algorithms

Information Flow Models 3

Unfortunately, extending Noninterference to include nondeterministic systems proved surprisingly difficult to pull off. As in the access control case, models proliferated

- ◆ Dondeducibility [Sutherland]
- ◆ Separability [Rushby]
- ◆ Noninference [O'Halloran]
- ◆ Generalized Noninference [McLean]
- ◆ Generalized Noninterference [McCullough]
- ◆ Restrictiveness [McCullough]

Probabilistic, timed models (e.g., PNI [Gray], FM [McLean], AFM [Gray]) were also developed, but they had their own issues

- ◆ Relevant timing information and probability distributions are not generally available at specification time

Where were we?

We had a “practically perfect” theory of security (Noninterference) that addressed the problem of covert channels but which was limited to deterministic systems.

We had many nondeterministic versions of Noninterference but there was no general theory that related these models and only ad hoc results with respect to important properties. The most important of these properties was composability, first brought to light by McCullough in 1987.

- ◆ Given systems $S1$ and $S2$ satisfying properties $P1$ and $P2$, respectively, what properties are satisfied by a composite formed by composing $S1$ and $S2$ with composition construction C ?
- ◆ Why are certain models composable and others aren't?

Digression: Alpern & Schneider

Alpern and Schneider framework consists of

- ◆ Infinite traces of system states
- ◆ Systems = sets of traces the system can exhibit
- ◆ Properties = sets of traces that satisfy a property
- ◆ Refinement: A system satisfies a property if its a subset

Within this framework, one can

- ◆ Define safety properties and liveness properties
- ◆ Prove that every property is the intersection of a safety property and a liveness property
- ◆ Prove the Abadi & Lamport Composition Theorem, which allows one to compute the behavior of a composite from the behaviors of its components

Why didn't this work for security (Rushby)?

What happened in 1994?

The reason that Alpern-Schneider doesn't apply to information flow properties is because they aren't sets of traces, but rather properties of sets of traces [First published by McLean but based on an observation by Jim Gray]

- ◆ Noninterference applies to a set of traces iff the presence of certain trace(s) in that set implies the presence of other trace(s) in that set

The Alpern-Schneider Framework can be extended to include security properties by the addition of Selective Interleaving Functions [McLean]

- ◆ Selective interleaving functions take two traces and return a third trace that is the result of interleaving the first two in a variety of ways depending on the type of the selective interleaving function
- ◆ Standard Noninterference properties can be viewed as closure properties of selective interleaving functions
- ◆ Composition theorems can be developed for these closure properties

Selective Interleaving System State

A *system state* is an ordered pair: $\langle \text{in}, \text{out} \rangle$

- ◆ $\text{in} = \langle \text{highin}, \text{lowin} \rangle$
- ◆ $\text{out} = \langle \text{highout}, \text{lowout} \rangle$

A *trace* is an infinite sequence of states: $\langle \text{state-1}, \text{state-2}, \dots \rangle$

- ◆ $\text{highin-}i$ = high-level input at time i
- ◆ $\text{lowin-}i$ = low-level input at time i
- ◆ $\text{highout-}i$ = high-level output at time i
- ◆ $\text{lowout-}i$ = low-level output at time i

A *system* and a *system property* are both sets of traces

- ◆ A system is the set of traces it can exhibit.
- ◆ A property is the set of traces that satisfy it.
- ◆ A system S *satisfies* (*refines*) a property P if $P \subseteq S$.

Selective Interleaving Functions

A function $f: \text{trace} \times \text{trace} \rightarrow \text{trace}$ is a *selective interleaving function* of type $F\langle \text{highin-select}, \text{lowin-select} \rangle \langle \text{highout-select}, \text{lowout-select} \rangle$ if and only if $f(\text{trace1}, \text{trace2}) = \text{trace3} \Rightarrow$

- ◆ $\text{highin-select} = 1$: $\text{highin}(\text{trace3}) = \text{highin}(\text{trace1})$
- ◆ $\text{highin-select} = 2$: $\text{highin}(\text{trace3}) = \text{highin}(\text{trace2})$
- ◆ $\text{highin-select} = 0$: $\text{highin}(\text{trace3})$ is **unrestricted**
- ◆ Analogous for lowin-select , highout-select , and lowout-select

-- Example --

Selective interleaving function of type $F\langle 1, 2 \rangle \langle 0, 2 \rangle$

	highin	lowin	highout	lowout
trace: t1	a1	b1	c1	d1
trace: t2	a2	b2	c2	d2
$f(t1, t2)$	a1	b2	*	d2

Information Flow as a Closure Property

Separability: Given any two traces, s and t , in some system S , there is a third trace in S with s 's high-level events and t 's low-level events.

- S is closed under the selective interleaving function of type $F\langle 1,2 \rangle \langle 1,2 \rangle$

Generalized Noninterference: Given any two traces, s and t , in some system S , there is a third trace in S with s 's high-level inputs and t 's low-level events.

- S is closed under some selective interleaving function of type $F\langle 1,2 \rangle \langle 0,2 \rangle$

Noninference: Given any trace, t , in some system S , the trace formed by setting all of t 's high-level events to λ is a trace.

- S is closed under the selective interleaving function of type $F\langle 1,2 \rangle \langle 1,2 \rangle$ where trace1 is $\langle \langle \lambda, \lambda \rangle, \langle \lambda, \lambda \rangle, \dots \rangle$

Generalized Noninference: Given any trace, t , in some system S , there is a trace in S with t 's low-level events, but whose high-level inputs are λ .

- S is closed under some selective interleaving function of type $F\langle 1,2 \rangle \langle 0,2 \rangle$ where trace1 is $\langle \langle \lambda, \lambda \rangle, \langle \lambda, \lambda \rangle, \dots \rangle$

Composition

External Constructs (e.g., for network construction)

- ◆ Product: parallel composition
- ◆ Cascade: sequential composition
- ◆ Feedback: sequential composition where the second system's output is directed back to the first system

Internal Constructs (e.g., for policy integration)

- ◆ Union
- ◆ Intersection
- ◆ Set Difference

Note that

- ◆ Product, Cascade, and Feedback span all compositions
- ◆ Intersection and set difference can be used to form refinements.

Example Composition Theorems

Product Theorem: $S1$ is closed under selective interleaving of type $F< IN1 > < OUT1 >$ and $S2$ is closed under selective interleaving of type $F< IN2 > < OUT2 >$ if and only if the product of $S1$ and $S2$ is closed under some selective interleaving function of type $F< IN1 : IN2 > < OUT1 : OUT2 >$

The corresponding theorem for Cascade is a bit trickier but it has some straightforward corollaries. Eg:

- ◆ If $S1$ satisfies property $P1$ and $S2$ satisfies property $P2$, where $P1$ is at least as strong as $P2$, then the cascade of $S1$ and $S2$ satisfies $P2$.

A Feedback Composition Theorem also has straightforward corollaries

- ◆ Separability is preserved when composed with itself via feedback
- ◆ Noninference is preserved when composed with itself via feedback
- ◆ If $S1$ satisfies Noninference and $S2$ satisfies Separability, then $S1$ composed via feedback with $S2$ satisfies Noninference

Is this science?

We started out with an almost perfect theory of security, but it was limited to deterministic systems.

We developed a grab bag of information flow security models that had various properties and no real understanding of what was going on.

Selective Interleaving Functions showed that these definitions could all be formulated in a common framework that

- ◆ systematized our knowledge of these properties
- ◆ gave us verifiable predictions about composition of these properties
- ◆ and explained why certain types of composition were harder on security than others
 - For example, the framework explained why general hook-up is hard on security properties. General hook-up includes feedback which requires the existence of a lot of traces for it to preserve a property.

Case Study Epilogue

Before moving on let me emphasize that there are many other excellent examples of science in the theory of computer security. To name just a few:

- ◆ Schneider's work on Enforceable Security Properties
- ◆ Clarkson & Schneider's work on Hyper-properties
- ◆ Abadi & Plotkin's work on Protection by Layout Randomization
- ◆ Many many paper on cryptographic protocol analysis, including the BAN Logic, Cathy Meadows' and Jon Millen's early formal analysis tools, etc
- ◆ Many papers on very high assurance system developments, such as cryptographic devices
- ◆ Etc, etc, etc

I picked the two I did, not because they are any better, but because I had first hand experience and they unfolded within the scientific community for computer security. For whatever reason, 1987-1994 seems to be an especially important era for that community.

Are we safe?

Limitations, warnings, and ways forward

Engineering

Note that the examples I provided of good science have all involved the theory of computer security.

However, the questions we wanted answered in a scientific way were about the security of systems.

- ◆ Even if we had a perfect science of security, security in practice is the designing, building, maintaining, and improving secure systems via the application of those principles – ie, it requires engineering.

We must accept the fact that scientific excellence does not guarantee engineering excellence.

- ◆ Security requires both.

Engineering 2

Engineering involves people, materials, processes, and products.

- ◆ Assuring people, materials, and processes is expensive and impractical in a global market, especially when we must worry about deception.
- ◆ We can apply science to the product, but this is different from applying science to a natural product.
 - The state space is discontinuous and very large
 - Some product behaviors (eg back doors) may be designed to be hard to detect.
 - The behavior of (intended and nonintended) users is hard to characterize and may be deceitful.
- ◆ Where we control these things (eg, as in the development and use of DoD cryptographic devices) we do quite well, but that is a very small subset of what we need to secure.
- ◆ And even then there are failures. Think of Enigma.

Where we can't control things, we need data and testing. That's where we most need improvement.

Security Testing

Scientific theories are evaluated by a repeatable process.

- ◆ Mathematical proof certainly provides validation, but not everything is provable and proofs are only as good as the community that vets them.
- ◆ The safety community calculates failure probabilities, but these assume independence of failures and so are hard to apply to security.

Examples of (why we need) good experiments

- ◆ Although a six digit cipher lock has a key space of 1,000,000 possible cyphers, at least one study suggests that if users can set their own combination, 6 digit dates (eg, 031714) are unusually common. Similar results hold for passwords generally. [Joseph Bonneau]
 - On the other hand, I haven't seen a study showing that randomly generated passwords are more secure if one takes into account the fact that they are more likely to be written down.
- ◆ Levla Bilge and Tudor Dimitras work on Zero-Day Attacks
- ◆ Ross Anderson's *Why Crypto Systems Fail*

Security Testing 2

Bottom line is that we need experiments to determine how well human engineered systems (where engineers may or may not be trustworthy) protect information in a environment of intelligent users/adversaries (authorized and trustworthy, authorized but untrustworthy, unauthorized) who can adapt and change behavior over time.

Examples

- ◆ Research has been going on for years on visually representing the state of a network's security, but I've never seen a study that evaluates how well a given visualization works for the purpose at hand.
- ◆ Things will only get worse when we start fielding intelligent security systems (eg, for attack detection and response) since they're behavior will be unpredictable by design.

Conclusion

We have some very good scientific work in the theory of computer security which show true science and a maturing scientific community.

- ◆ I've looked at two examples, but there are many others.

That said, we must accept the fact that scientific excellence does not guarantee engineering excellence.

- ◆ Security requires both.

We are starting to develop some experimental work to gather real data about what we cannot control, but it lags behind the theory

- ◆ One of the biggest hindrances is that we're designing systems that will be operated by humans and will be attacked by systems supplemented by human intelligence. Humans understand human failings.
- ◆ As is the case in the case of developing biological vaccines, the attackers will modify behavior over time.

Thank you.