Isolation and Flow of Information

CMMRS 2019 (2/3)

Saarbrucken, Germany
August 2019

Fred B. Schneider
Samuel B Eckert Professor of Computer Science

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

Cornell CIS
Computer Science

Security Trade-offs: Inconvenience

Approaches to enforcement:

- Monitoring.
  - *Authenticate source of each request*
  - *Use context of past action to block future violations*

  Inconvenience: Authentication of users, authentication of programs, limitations on flexibility
- Isolation.
  - *Keep attackers out or keep attackers in.*

  Inconvenience: Blocks communications between programs.

Security Trade-offs: Values

Enforcement can be in conflict with

- Privacy
- Openness
- Freedom of expression
- Opportunity to innovate
- Access to information

Values differ across jurisdictions in Internet.

Build on the Past?

Flawed analogies lead to flawed interventions.

- Liability lawsuits
- Insurance to limit exposure (and transfer risk)
- Deterrence through accountability

Flawed Analogies: Liability

Basis: Comparison of observed performance with some basis for acceptable behavior.

- Need a specification
  - Expensive to produce
  - Limits extension (functionality or environment)
  - Weak specifications do not rule out attacks.
  - Strong specifications rule out re-purposing

Flawed Analogies: Insurance

Basis: Data about past incidents and payouts used to predict future payouts (and determine price).

- Software evolution is discontinuous
- Changes to environment are uncontrolled by user, developer, insurer.

Flawed Analogies: Deterrence

Basis: Identify attackers and punish them.

- Attribution of cyber-attacks difficult
- Jurisdiction of attacker might not be willing to help.

Balkanize the Internet into regions of cooperation delimited by monitoring?

Packaging to Create Incentives

Cybersecurity doctrine:

- **Goals** define
  - kinds and levels of cybersecurity sought
  - acceptable trade-offs and costs.
- **Means** include
  - Technical / education / regulation
    - Incentives: market-based to coercive

A lens for viewing policy proposals.

Early Doctrine:
Doctrine of Prevention

Build systems that don't have vulns.

- Unworkable:
  - Big systems are too complicated to get right.
  - Formal verification infeasible
  - Exhaustive testing infeasible
  - Performance standards would require security metrics.
- Incomplete:
  - Ignores users and operators ("social engineering")
  - Environment not static (attacks, assumptions, uses)
    - Specifications must evolve
    - Assurance argument must be reconstructed

Early Doctrine:

Doctrine of Risk Management 1

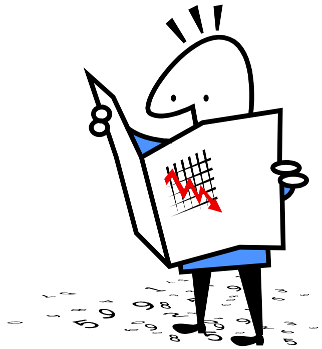
Invest in security to reduce expected losses due to attacks.

- Cost of attack
  - What is value of confidentiality? Integrity?
  - What is the cost of recovery from attack?
  - What about costs to third parties?
- Probability of attack
  - Insufficient data about threats and vulns.

Early Doctrine:

Doctrine of Risk Management 2

- Under-investment is rational.
  - Individuals cannot:
    - reap full benefit from their investments.
    - cannot control vulns.
  - No metrics to predict ROI
  - Insufficient data about threats, vulns, and cost of losses
  - Continuing investments would be needed
    - Threats co-evolve with defenses
    - Replacement systems and upgrades constantly deployed
    - New domains mean new forms of security needed.



Recent Doctrine:
Doctrine of Accountability

Deter attacks through threats of retribution.

- Retrospective and punitive
  - No concern about keeping systems up and running.
- Attribution of action is often infeasible.
  - Cross border enforcement?
  - Non-state actors?
  - Binding of machines to individuals is weak.
- Incomplete:
  - Narrow set of policy options for privacy.
  - Presumes attacks are crimes.

Toward a New Doctrine:

Public Goods

Thesis: Cybersecurity is a **public good**.

- Non-rivalrous: Consumption of the good by one individual does not reduce availability for consumption by others.
- Non-excludable: No individual can be excluded from having access to the good.

"Public health" is a public good, too...

Public Health?

… duties and power of the state to assure health of the population (not individual) and limitations on that power to protect the interests of individuals.

- Herd immunity vs individual vaccination risk
- Stem an epidemic vs individual privacy
- Incentives vs externalities

Doctrine for Public Health

Goals: Prompt production
Manage its absence

Means: Education, prevention, surveillance, containment (quarantine), diversity, mitigation, recovery.

- Eschew: punishment, compensation, restitution

Requires new research and always will.

- Pathogens evolve.
- Expectations and health needs grow.

Public Health → Public Cybersecurity

- Network: people → computers (+ people)
- Positive state: health → cybersecurity
  - Produce: health → produce cybersecurity
  - Manage: disease → manage insecurity (vulns)

Public Health → Public Cybersecurity

- Network: people → computers (+ people)
- Positive state: health → cybersecurity
  - Produce: health → produce cybersecurity
  - Manage: disease → manage insecurity (vulns)

Doctrine(s) of Public Cybersecurity:

- ***Prompt the production of cybersecurity.***
- ***Manage the remaining insecurity.***
- ***Political agreement to balance individual rights and public welfare***

Public Cybersecurity Mechanisms

- Produce Security
- Manage insecurity
  - Diversity (obfuscation/randomization)
  - Monitoring
    - Boundary traffic-monitoring (firewalls, Einstein)
    - Mandate ISP coordination
  - Patching (cost subsidy, injured party comps)
  - Isolation (vs encryption, vs censorship)
  - Intermediaries (ISP's)

Important Metaphors

- Cyber-attacks as crime
  - Deterrence through Accountability
- Cyber-attacks as disease
  - Public Cybersecurity
- Cyber-attacks as warfare
  - ???

One defense suffices: Avoid vulnerabilities.

For additional information

Doctrine for Cybersecurity Deirdre K. Mulligan and Fred B. Schneider. In *Daedalus, Journal of the American Academy of Arts & Sciences*, Fall 2011 (“Protecting the Internet as a Public Commons”, eds. David D. Clark and John B. Horrigan). Pages 70-92.

Impediments with Policy Interventions to Foster Cybersecurity Investment Fred B. Schneider. *Communications of the ACM* Vol 61, No. 3 (March 2018), 38—38.

Produce Security:
Enforcement Strategies

- Isolation
- Monitoring
- Recovery
- Asymmetric Computation

Walls Enforce Isolation

- Prison walls: Keep people in.
- Fortress walls: Keep people out.
- Windows/doors allow activities on one side to influence the other side.
  - Holes degrade isolation.
  - Holes are not always apparent.
    - acoustic, energy, timing

Isolation in computing systems

Allows stronger assumptions about a component's environment: less need to trust the environment.

- Restrict environment form changing component's state
  - Protects **integrity**
- Restrict environment from influence by component
  - Protects **confidentiality** of sys state

Units of Isolation

- Physical isolation
- Processes and virtual machines
  - Mapping
  - Time multiplexing
- Measured principals
  - Cryptography

Mapping for Isolation

Environment for a process:

- Instruction set. Uses “names” for variables.
- Memory. Associates values with names.

Idea: Interpose per-process mappings map.P:

map.P: names → values

by loading register MR with map.P.

Enforce disjoint ranges for mappings.

Process Switch = Mapping Switch

While executing P: MR = map.P

To start executing Q: MR := map.Q

Implementation details

- Map.P implemented by pairs { … <*n, addr*> … }
  - Limit possible mappings to enable smaller tables
    - <*n, lim, addr*> maps: *n+A*   to   *addr+A* **if** *A* < *lim*
  - Only "trusted" software executes "MR := … "
    - Impl by having processor modes: **system** versus **user**
    - Only trusted software executes in **system** mode.

Time multiplexing for isolation

Interposition of mapping not always available.

From time t to t′: P has exclusive access to r.

Otherwise: state of resource r saved for P in R[P]

Transitions:

- Instructions to cause: r := R[P] or R[P] := r
- Interrupt to cause: R[P] := r

Isolation by Cryptography

- Encryption can enforce confidentiality
- Digital signatures can restrict updates.
  - N.b. Unauthorized writes destroy availability

But need protection for cryptographic keys! Soln:

- Generate and store keys in special registers.
  - Assumes tamper-proof hardware
- Execute cryptographic functions in hardware.
- Control access to cryptographic functions.

Measured Principals

Abstractions used in TPM, SGX, …

Measured Principal: Properties of its execution can be deduced from it's name. *Name is basis for trust*. Name is not just aspirational: "Windows 9"

Gating Function: K-F(…) is instantiated with a fixed config constraint C(K,F)---a set of names of measured prins allowed access. K is a key. F is a cryptographic function.

Names from Descriptions

N(D) is name *inextricably* linked to a **description** D

- D = <d1, d2, … dn> gives **descriptors** di (in order accessed for resources.
  - Each di depends on state and capabilities of resource.
- D allows predictions of what N(D) does.
- Modified description D’ gives modified name N(D’)
  - Gating functions deny accesses by modified name.
    - Patches and revision?
    - Access linked to initial state, not current state of resource.

Properties of Names

Prevent attackers from computing specific names.

= Prevent attackers from getting access to a gating function.

Name N(D) for measured principal should satisfy:

- $\neg(D = D')$ implies $\neg(N(D) = N(D'))$
- Infeasible to construct D′ where
  - $\neg(D = D')$ and $N(D) = N(D')$

Names as Hashes

Hashes have the required properties.

hc:= 0
for i := 1 **to** n
   hc := Hash(hc $\circ$ Hash(di))

Hashes can be computed incrementally

- Next resource computed during execution
- Past resources deleted (as in boot)

Descriptor Details

Descriptor should depend on contributions of resource to execution semantics for measured principal.

- Need: Equality test for descriptor vs resource
- Need: Equality implies equivalent behavior.
- Do not need: Transparency or property inference.

Might also have “hint” to identify actual resource to analyze for comparison.

Descriptors for Storage

Depends on values stored:

< dev,

strt, fin,

H(dev[strt] , dev[strt+1] … dev[fin] >

Descriptors for Interpreters

Implemented in software: Use descriptor for storage containing interpreter.

Implemented in hardware: Need unique id. HW cannot reveal id or else impersonation becomes possible using VMM.

Descriptors for HW Processors

- HW instance id has unique (private) signing key k.id
- HW has instruction to produce signature with k.id
- Verification (public) key K.id:
  - K.id used as that name of processor
  - HW read-only register could contains K.id
  - Certificate:

    Intel **says** K.id **speaks for** Intelx86

To check name: HW signs a challenge "r".

Gating Functions in Hardware

Sensible to trust hardware implementation of gating functions if:

- can trust manufacturers documentation
  - Certificate with name indicates documentation to read
- physical access to device required for compromise
- attackers do not have physical access to device

HW Features (simplified impl)

New instructions to

- Set (“extend”) configuration registers. Reset by reboot.
- Set key registers.
  - sealing key registers, quoting key registers, unbinding key registers.
- Compute gating functions:
  - Sealing (protects confidentiality and integrity)
    - Shared-key encryption
  - Quoting (establishes authenticity)
    - Public key decryption / digital signature
  - Binding (import remote content)
    - Private key decryption

Security Case for Keys

- Keys are created in key registers and never leave registers (unless encrypted).
- Instructions that read a key register KR do not reveal contents of KR.
- Keys remain in key registers after reboot but cannot be accessed unless configuration registers set again (but that requires loading trusted software).

Configuration constraints

Basis for allowing access to a gating function.

- Configuration constraint C is a set of pairs:
  - Name of configuration register CRi
  - Value in the configuration register CRi
- Configuration constraint is satisfied (or not) based on current values in configuration registers.
  - Implicit definition for set of measured principals

Loading configuration registers:

CRreset(CRi): CRi := 0

CRextend(CRi , mem): CRi := Hash(CRi * Hash(mem))

Gating Functions: General

KRgen(KRi, crSet) / KRgen(KRi, mem, crSet):

stores fresh key in key register KRi

(may also store certificate in memory mem)

crSet is set of pairs: < CRi , value in Cri >

F(KRi , in, out):

out := F(Kri, in) if crSet satisfied.

Gating Functions: Sealing

seal(KRi, in, out);

- Creates K/C-sealed bit string if C holds.

unseal(KRi, in, out);

- Given K/C-sealed bit string, retrieves original value. Executes only when C holds, KRi contains sealing key, and "in" unaltered since seal invoked.

Useful to store state between activations of a system, but protocol needed for software upgrade.

Gating Functions: Quoting

- Digital signature attests to configuration constraint at time of signing.
- Configuration constraint can be basis for trust in message contents.

KRgetConf(KRi, r, out):

- generates certificate with configuration constraint associated with key register KRi.

KRgetCurConf(crSet, r, out):

- generates certificate with configuration registers in crSet.

Gating Functions: Unbinding

Decryption with private key k "binds" information to configuration constraint Ck.

Remote host uses public key K to ensure content becomes visible only to systems satisfying Ck.

Awkward HW Abstractions

- Clients share a single set of configuration and key registers. So clients must trust each other.
  - Better: Implement per-client register sets

- Small set of cryptographic functions available as gating functions.
  - Better: Allow additional gating functions.

Per-client Registers

Instructions that access configuration and key registers are system mode.

- Instructions exist to save/restore KR's.
- No instructions to save/restore CR's.

So standard recipe to multiplex registers fails.

Build sw emulator for instructions and registers.

- Use seal/unseal for save/restore sets of KR's.
- Use software copy of CR's. If satisfied, invoke HW gating function where CR's are satisfied by emulator.

Remote Attestation

Protocol for R to obtain (for p on remote host S):

- Description Dp where N(Dp) = p
- Public key Katt.p
- Certificate: Katt.p **speaksfor** p

Use of Measured Principals

Applications of measured principals:

- Cloud services. No need to trust cloud operator.
- Digital rights management (IP). Prevent theft of digital content.
- More secure desktop. Prevent modification to run-time.

Abuse of Measured Principals

System designer can prevent adding extensions.

- Lock-out competitor's products
- Impinges on computer owner's rights

Rights and responsibilities of being in an ecosystem:

- Keep system patched
- Prevent malware from being loaded