Computing in a Dangerous World

Howard Shrobe

Program Manager

DARPA/I2O

THE SITUATION

"60 Minutes"

November 8, 2009

"Several prominent intelligence sources confirmed that there were a series of cyber attacks in Brazil: one north of Rio de Janeiro in January 2005 that affected three cities and tens of thousands of people, and another, much larger event beginning on September 26, 2007.

That one in the state of Espirito Santo affected more than three million people in dozens of cities over a two-day period, causing major disruptions. In Vitoria, the world's largest iron ore producer had seven plants knocked offline, costing the company $7 million. It is not clear who did it or what the motive was.

But the people who do these sorts of things are no longer teenagers making mischief."

http://www.cbsnews.com/video/watch/id=5578986n&tag=contentMain;cbsCarousel

Cyber Physical Systems Under Attack

Bits (NY Times)

Business ▪ Innovation ▪ Technology ▪ Society

September 24, 2010, 8:41 PM

Malware Hits Computerized Industrial Equipment

By RIVA RICHMOND

… Security experts say Stuxnet attacked the software in specialized industrial control equipment made by Siemens … the first such attack on critical industrial infrastructure that sits at the foundation of modern economies.

Eric Chien, the technical director of Symantic … said it appeared that the malware was created to attack an Iranian industrial facility. … The specific facility that was in Stuxnet's crosshairs is not known, though speculation has centered on gas and nuclear installations.

… malware experts say it could have been designed to trigger such Hollywood-style bedlam as overloaded turbines, exploding pipelines and nuclear centrifuges spinning so fast that they break. "The true end goal of Stuxnet is cyber sabotage. **It's a cyber weapon basically**," said Roel Schouwenberg, a senior antivirus researcher at Kaspersky, a security software maker.

Wrecking a Generator Remotely

Increasing Malicious Cyber Activity

"If these trends continue through the end of 2009, there would be a 60 percent increase in malicious cyber activity compared to 2008. … in just the preceding six months, the U.S. military alone had spent more than $100 million … to remediate attacks on its networks"

2009 report to Congress of the U.S.-China Economic and Security Review Commission One Hundred Eleventh Congress, November 2009

Figure 1: DoD Reported Incidents of Malicious Cyber Activity, 2000–2008, With Projection for 2009

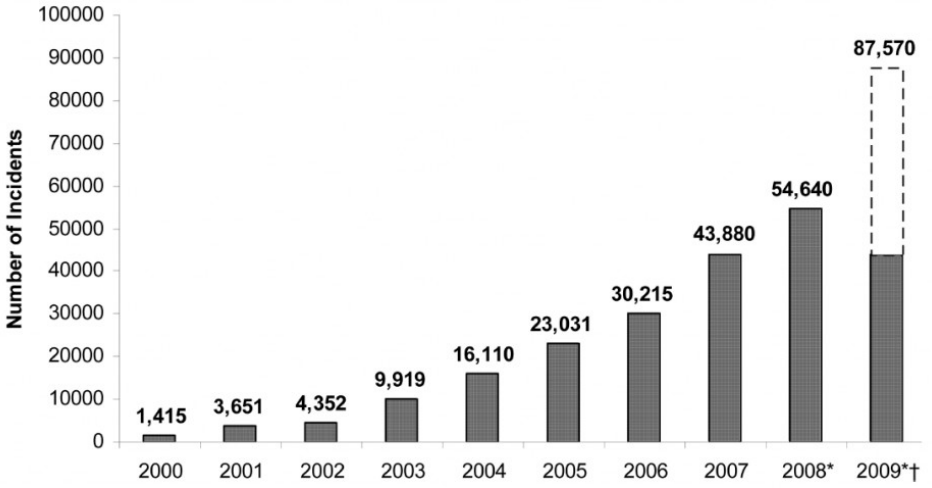
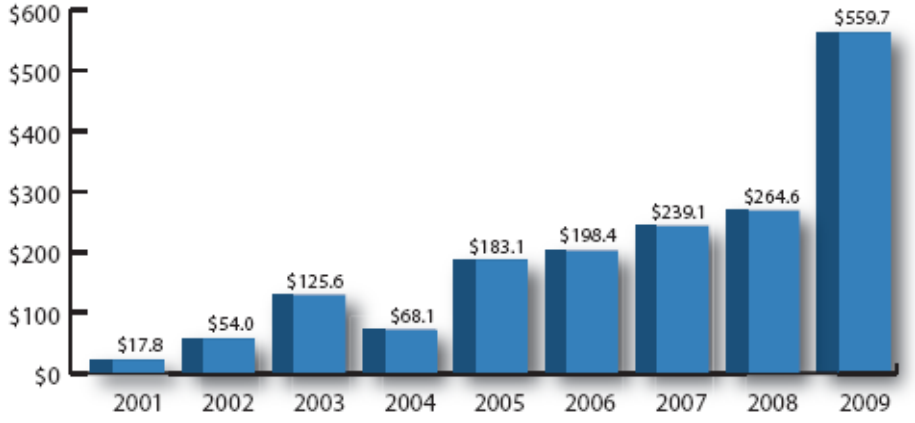
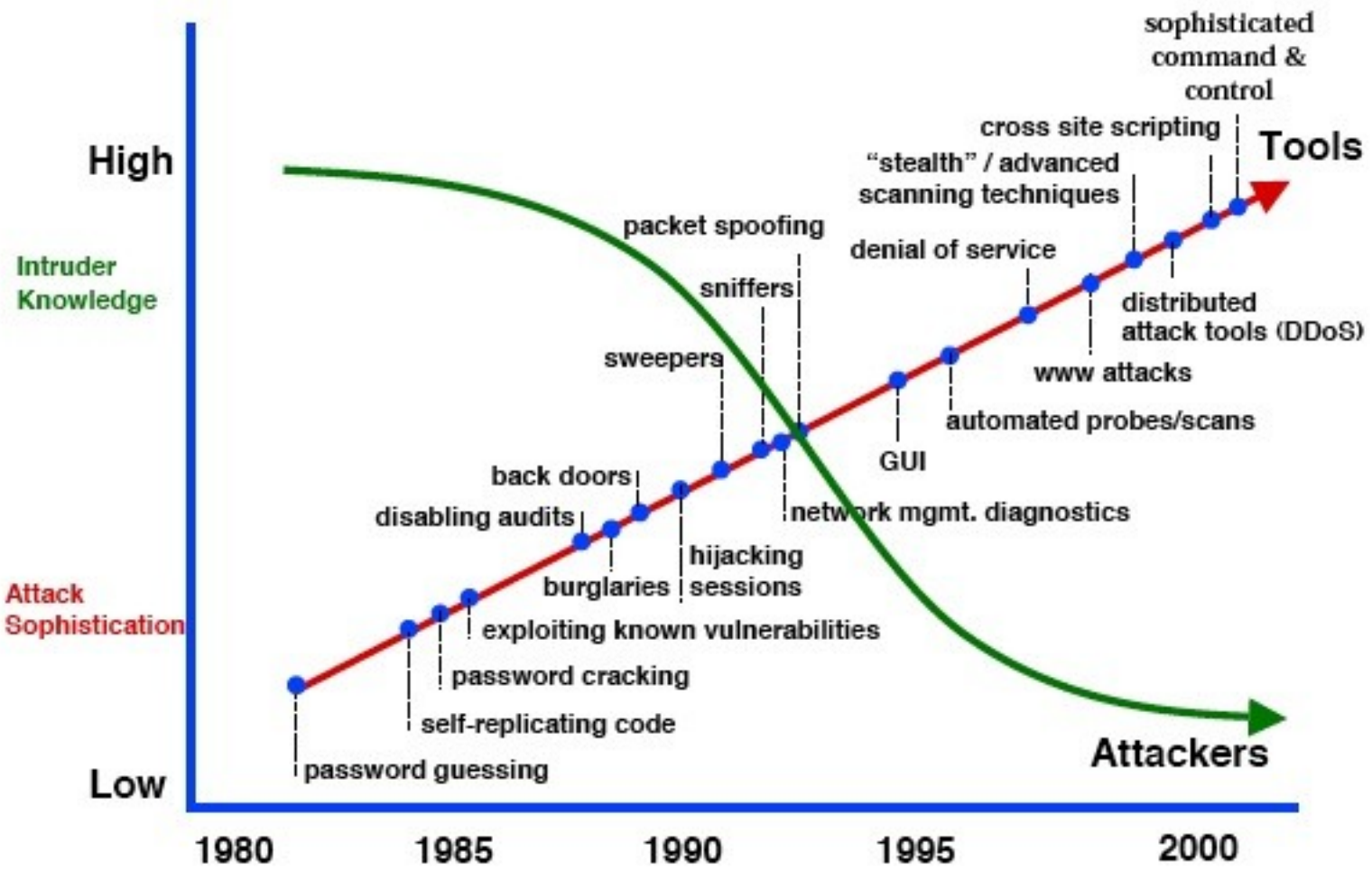


Figure 2: Yearly Dollar Loss (in millions) of Referred Complaints

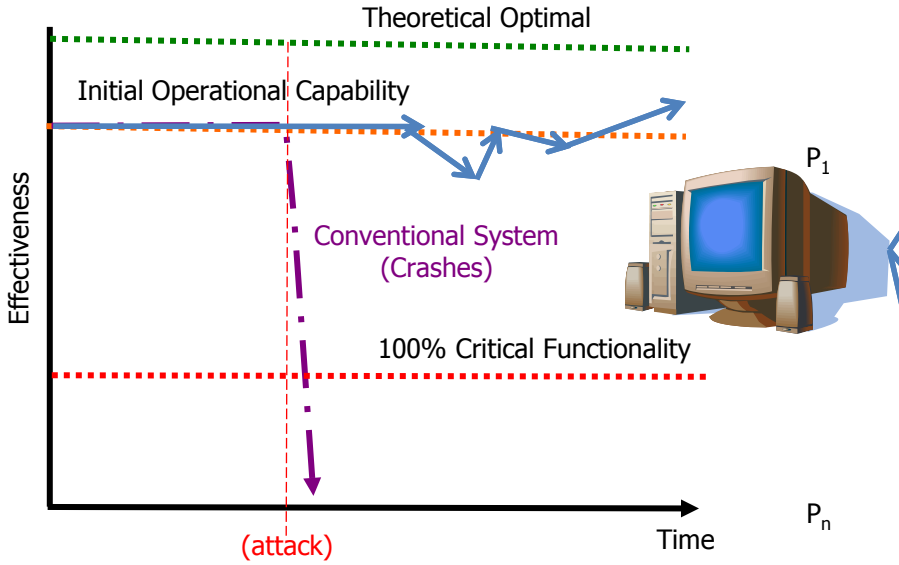




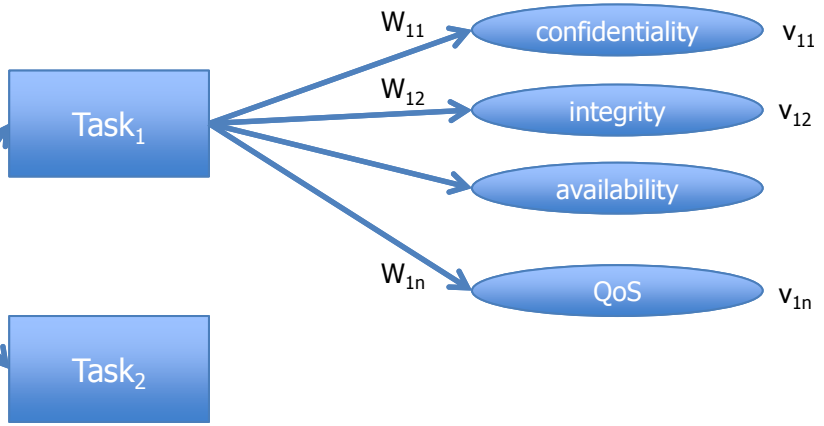
FRAMEWORK AND ANALYSIS
HOW DO YOU DEFEND AGAINST AN ORGANIZED THREAT?

Mission Effectiveness

The objective is to sustain **mission effectiveness**. Different mission components have different security needs and will make different trade-offs at run-time between these, quality of service, and even correctness.



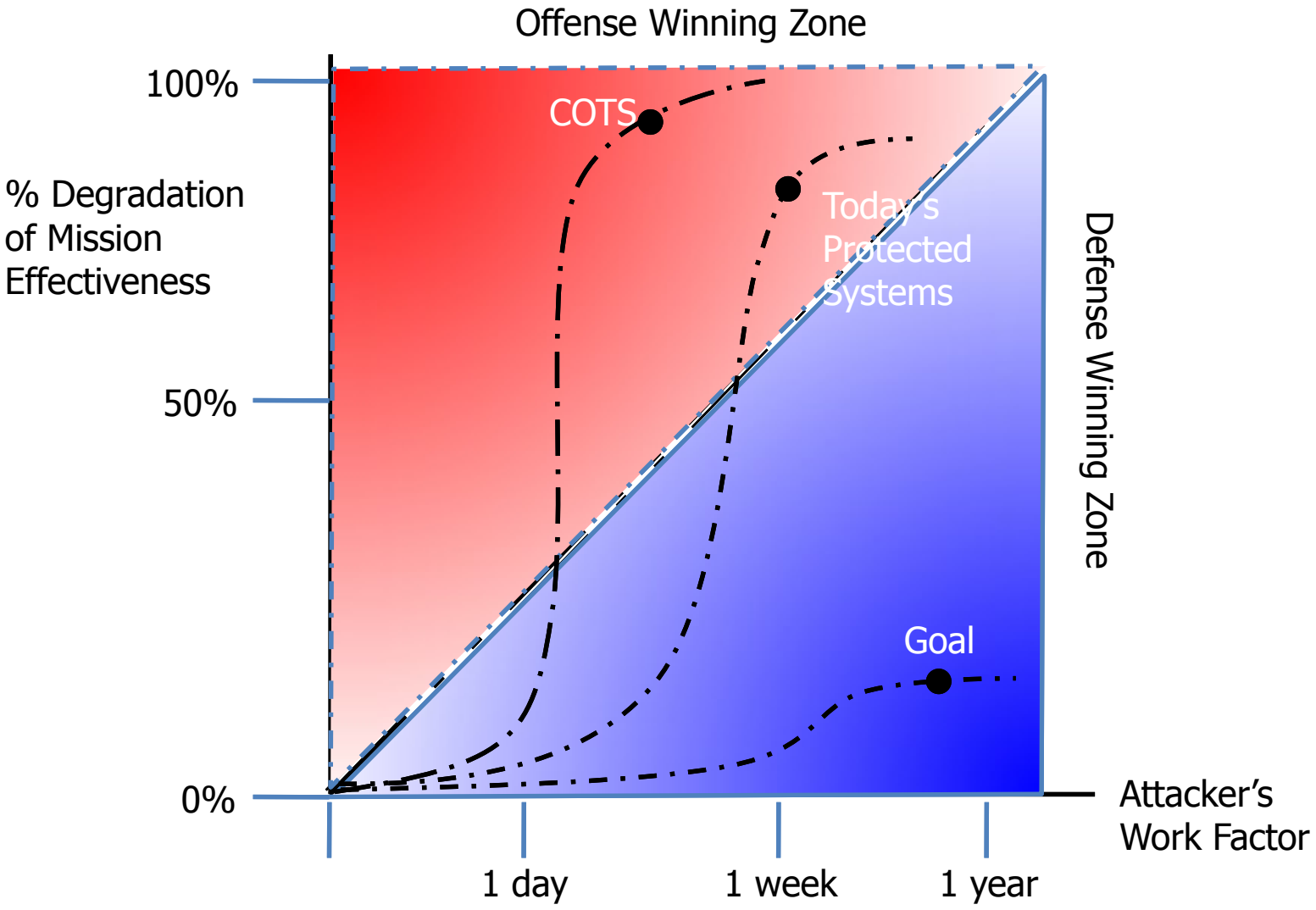
Metric for Mission Effectiveness



P_i = Priority of Task ***i***
W_{ij} = Importance of property ***j*** to Task ***i***
V_{ij} = Degree to which property ***j*** is achieved for Task I

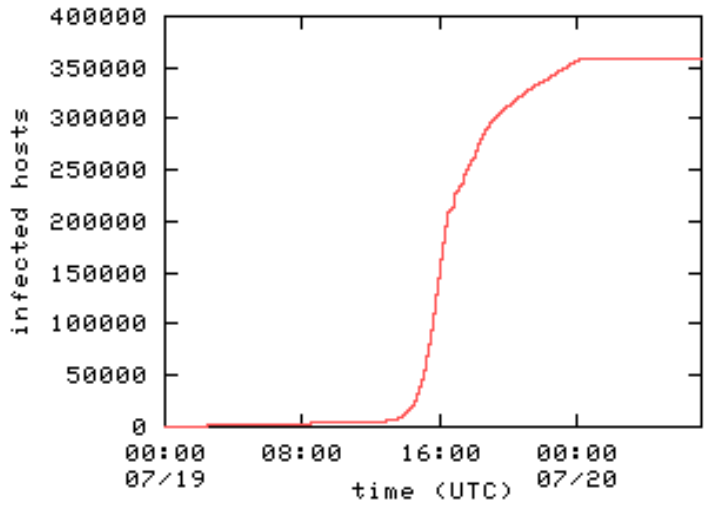
$$\sum_{i} \sum_{ij} * \,'_{ij}$$

Economic Tradeoff: Workfactor and Degradation

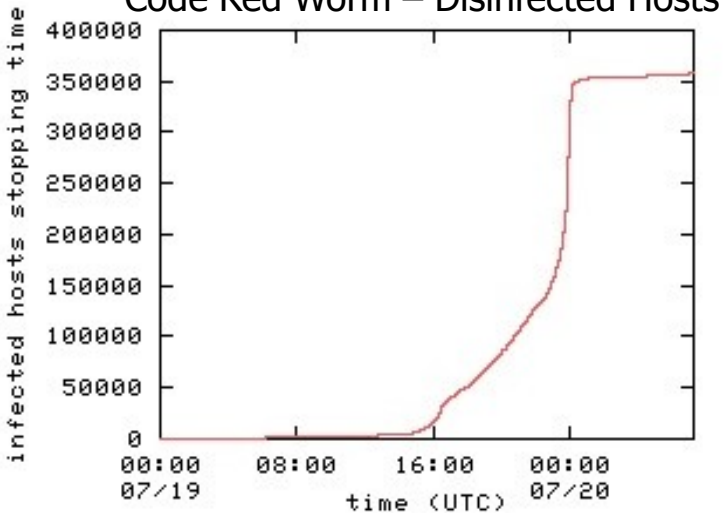


Attack and Healing Dynamics

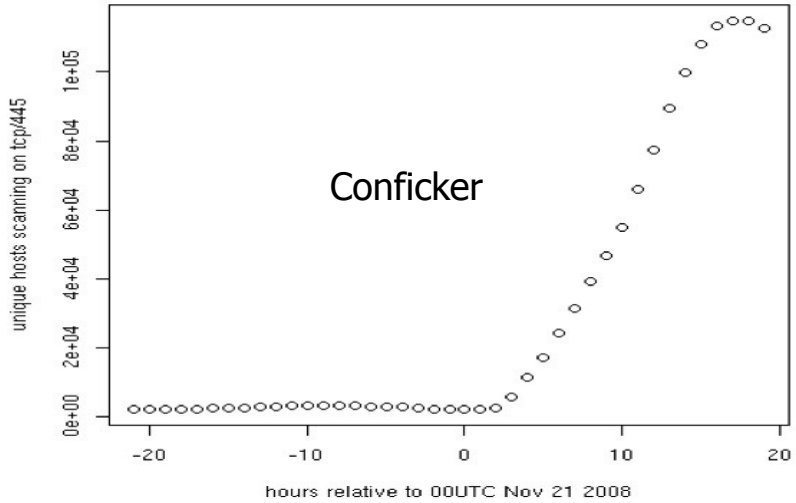
Code Red Worm – Infected Hosts



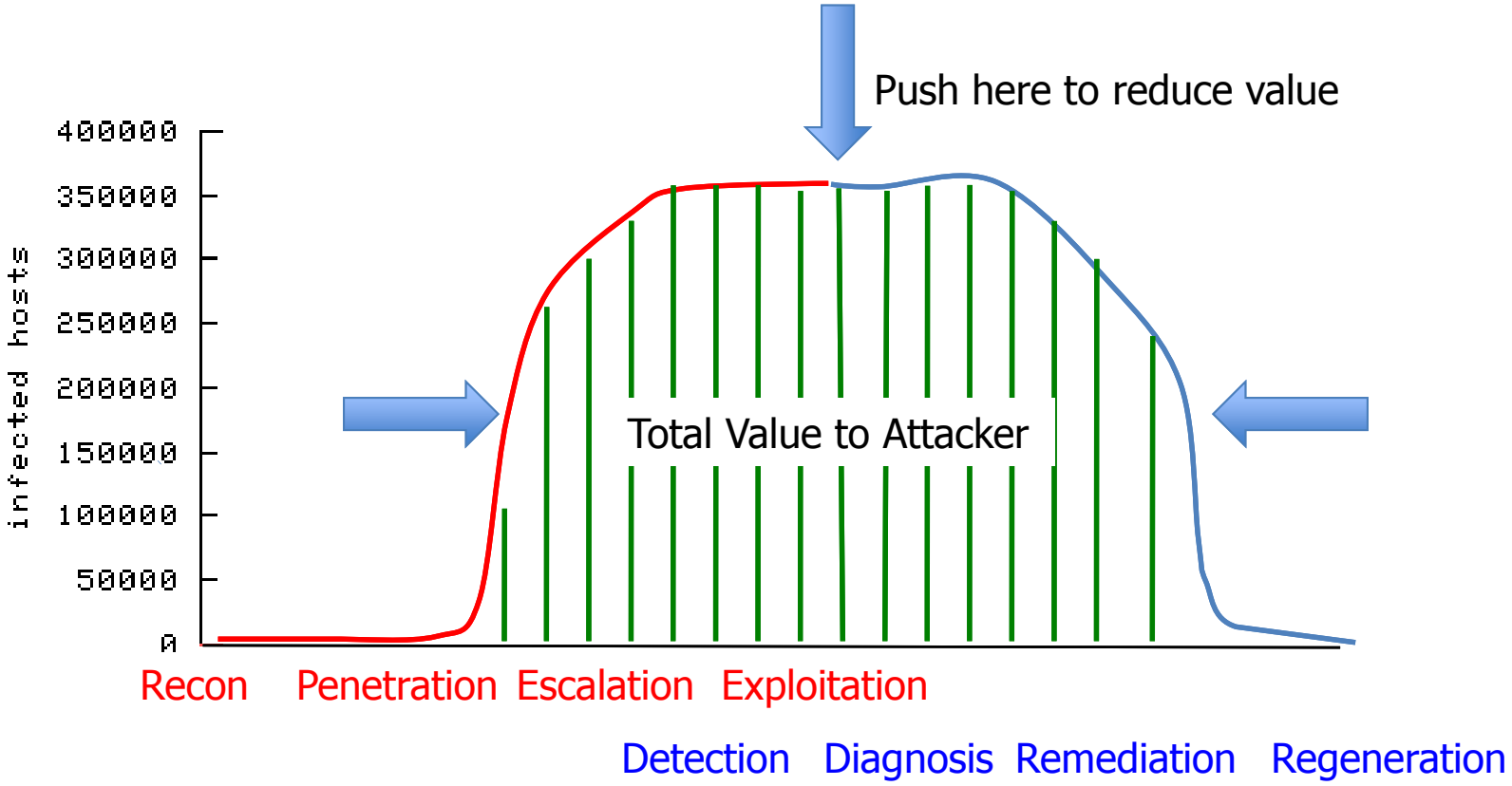
Code Red Worm – Disinfected Hosts



Conficker



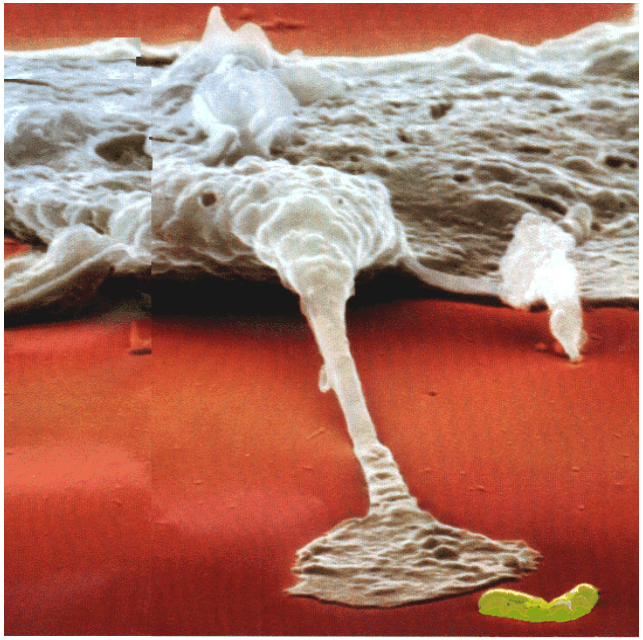
Total Value to the Attacker



Two Models of Survivability

Fortress (Traditional)

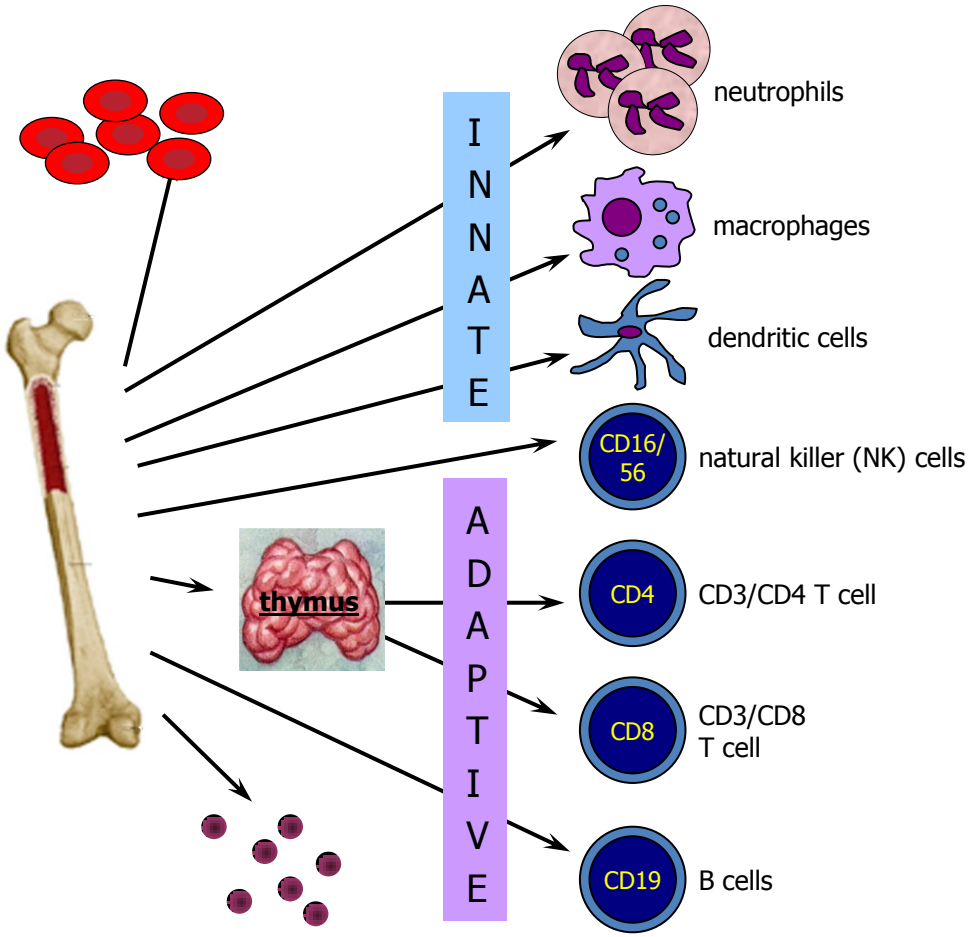
- Impenetrable (hopefully)
- Monolithic
- Single layer
- Rigid
- Immobile



Organism

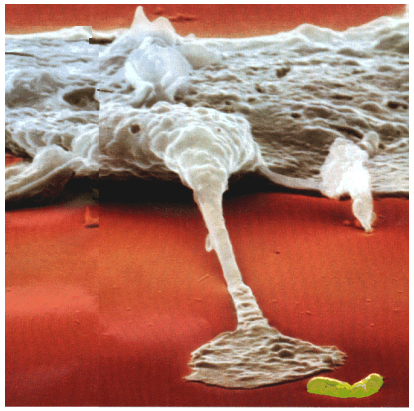
- Many partial barriers
- Heterogeneous
- Defense in depth and self healing
- Adapts, learns, evolves
- Mobile

Humans Have Two Immune Systems: Innate and Adaptive



Fast, but inflexible, covers fixed sets of pathogen that are always present. Supports the adaptive immune system.

Slower, learns to recognize new sets of pathogens, distinguishes self from non-self, retains memory to guard against future attacks.



At least 20 – 30% of the body's resources are involved in constant surveillance and containment.

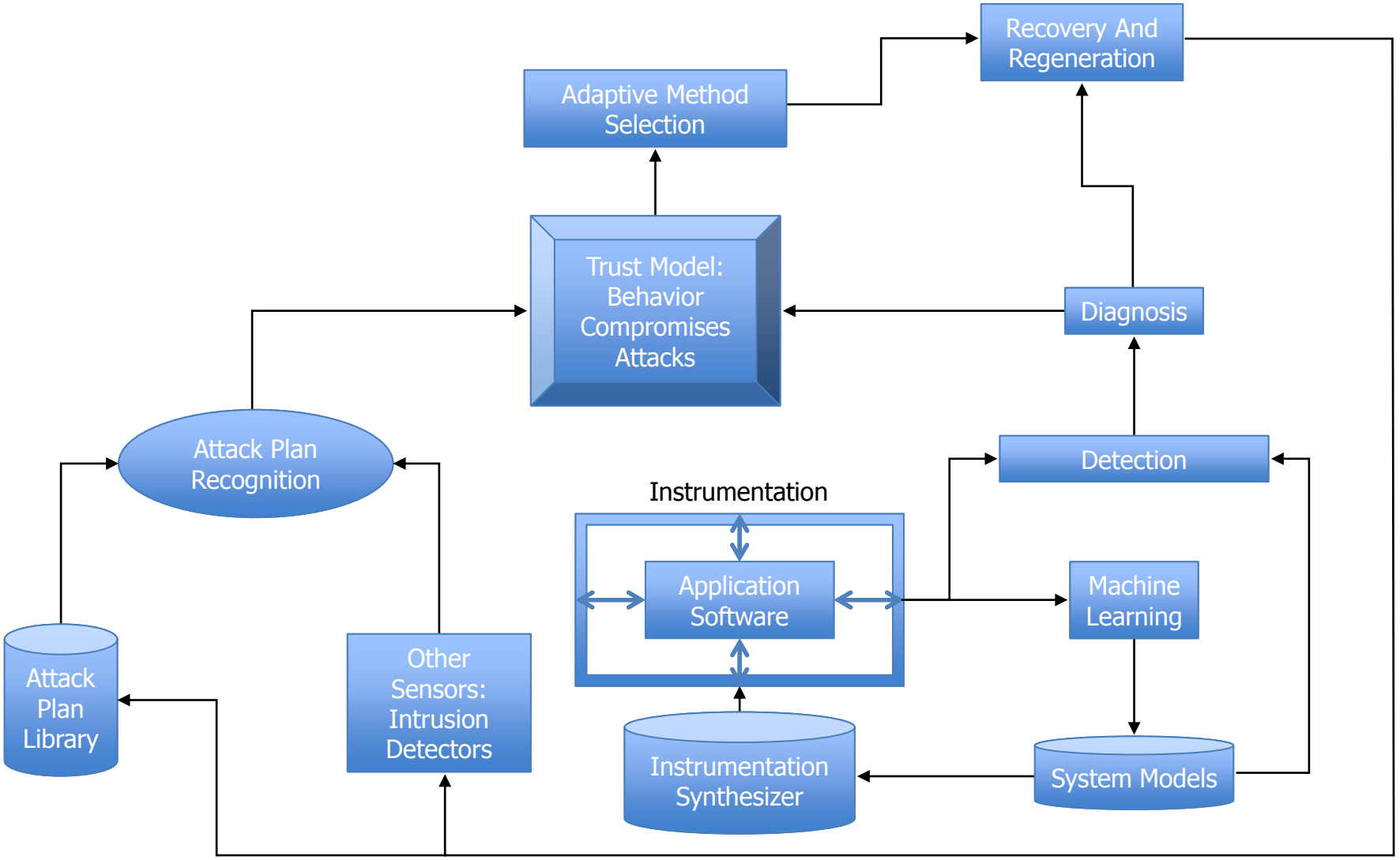
SELF-ADAPTATION AS A DEFENSE

Defensive Moves

- **Evasion and Obfuscation:** Be a moving and hard to understand target
- **Containment:** No "implicit trust" between system components
- **Diversity:** Be different from your peers
- **Detection:** Notice that things have gone wrong
- **Diagnosis:** Figure out precisely what caused things to go wrong
- **Recovery:** Continue functioning with graceful degradation even though some things might be damaged
- **Repair:** Get back to full functioning
- **Repulse:** Learn how to deflect similar attacks before they happen again (what doesn't kill you makes you stronger)

Who watches the watchman?

Defensive Architecture

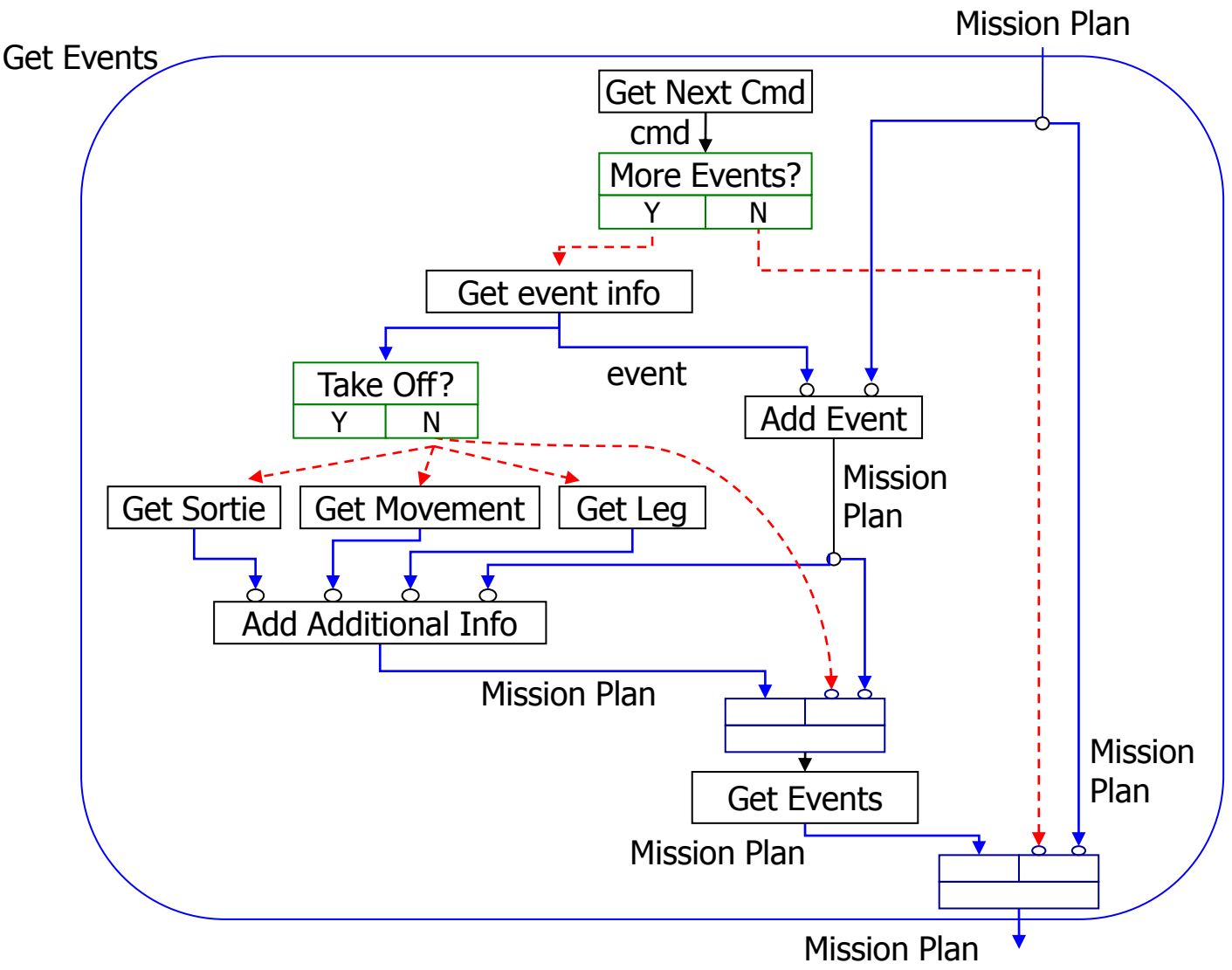


What Do System Models Include?

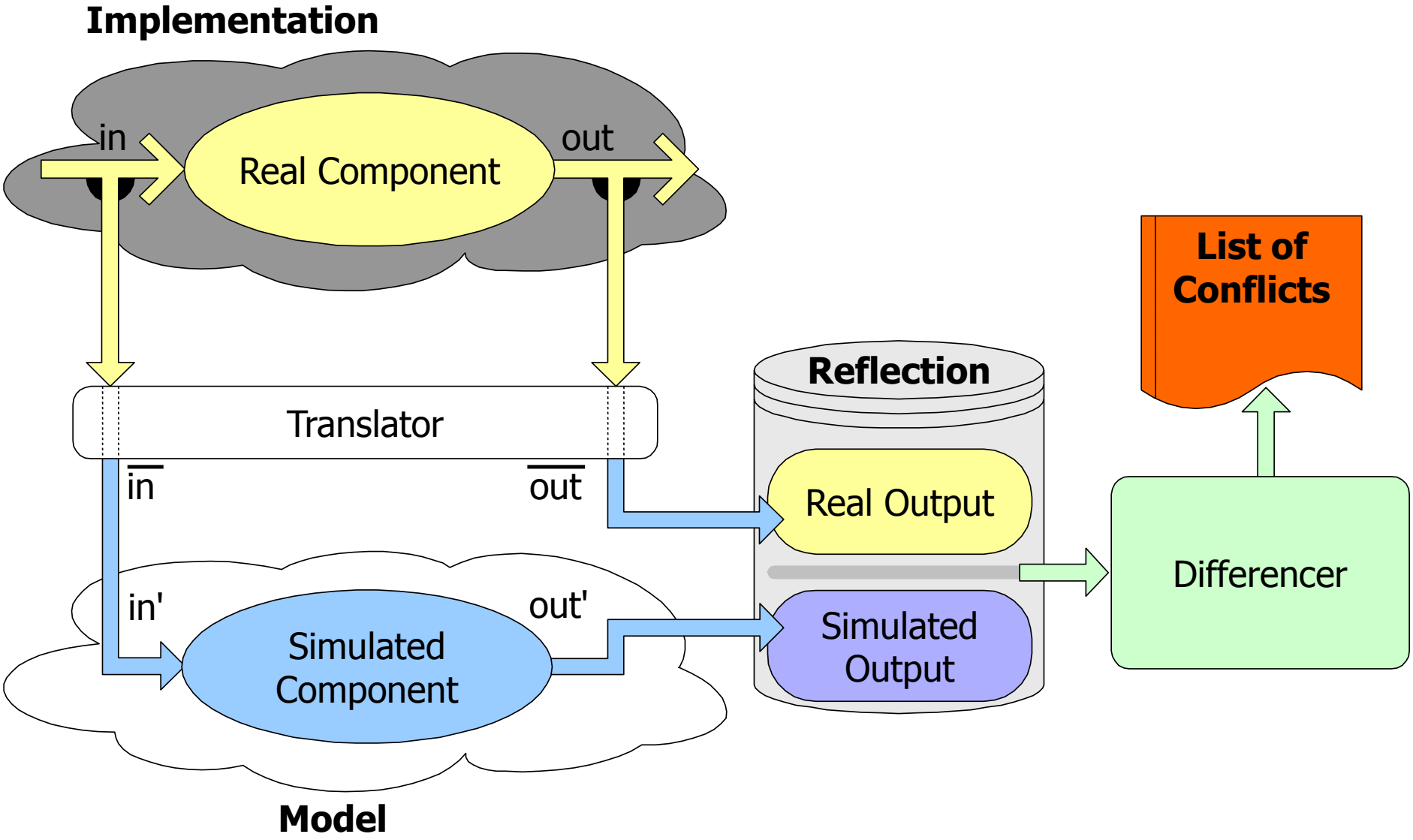
- Components
- Branches
- Joins
- Control Flow
- Data Flow

Annotated with:

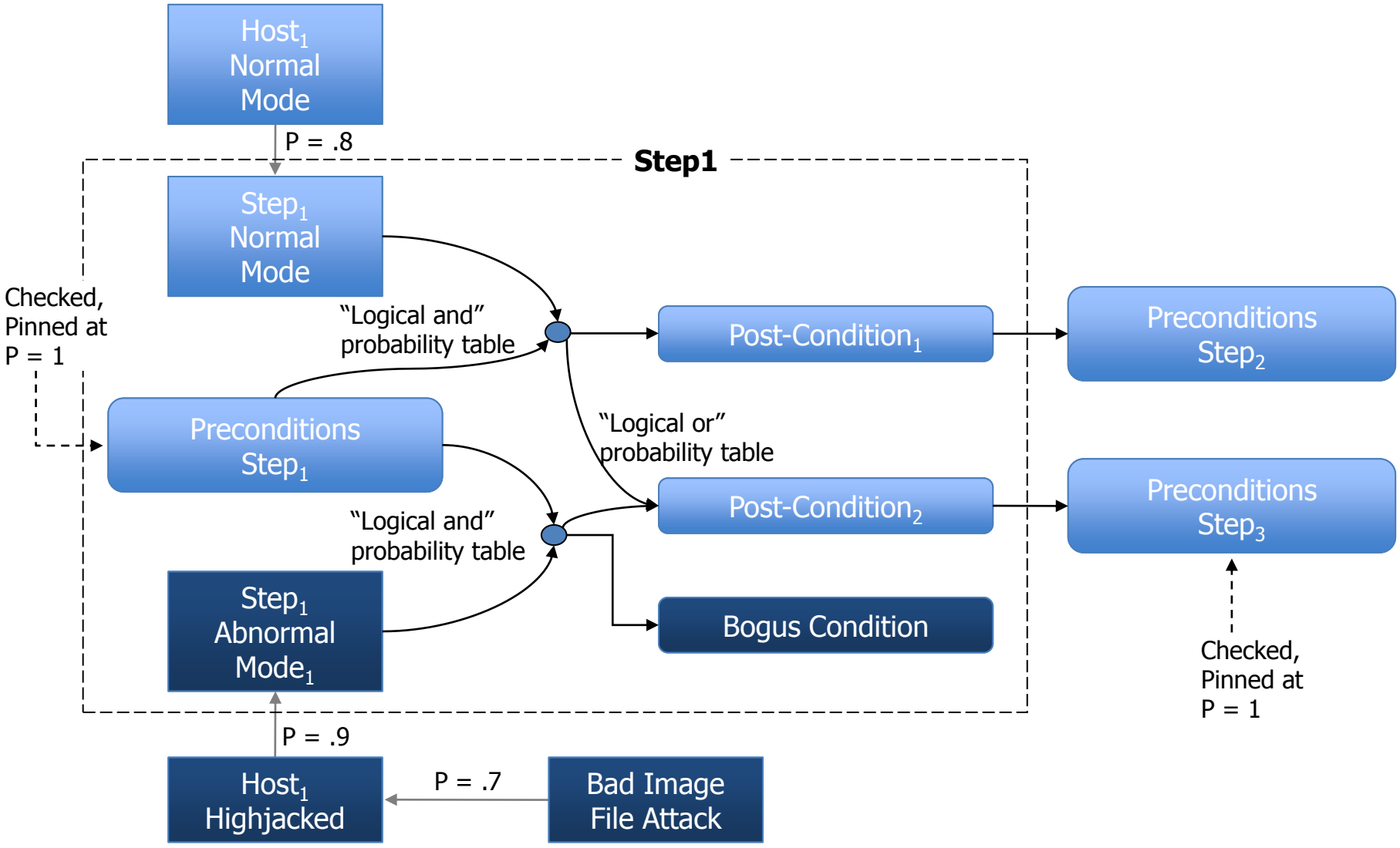
- Preconditions
- Post-conditions
- Across invariants
- Across prohibitions
- Invariants



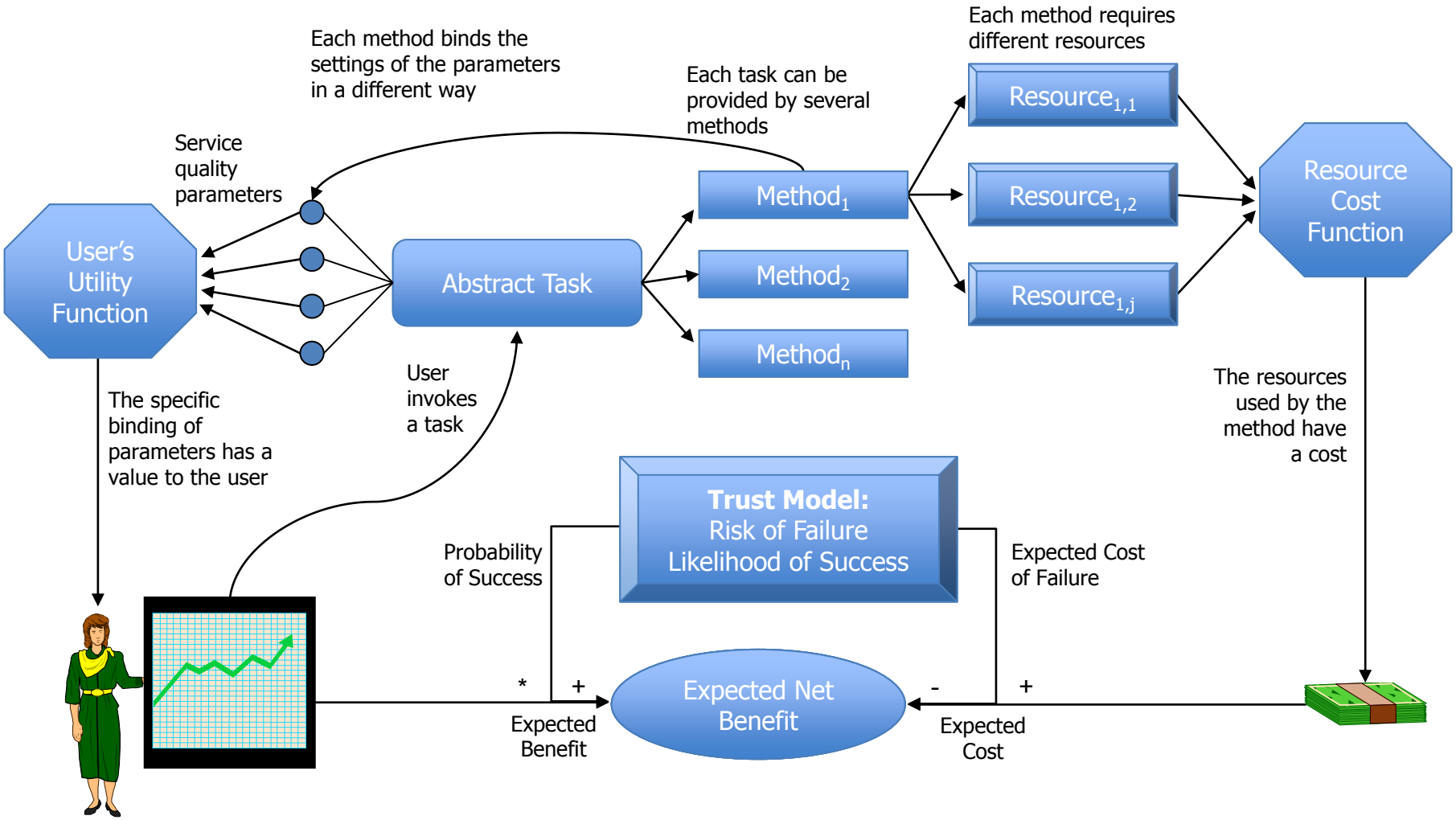
Detection by Architectural Differencing



Dependency Tracking Facilitates Diagnosis of Upstream Failures



A System Can Adapt by Selecting among Multiple Methods for Each Task



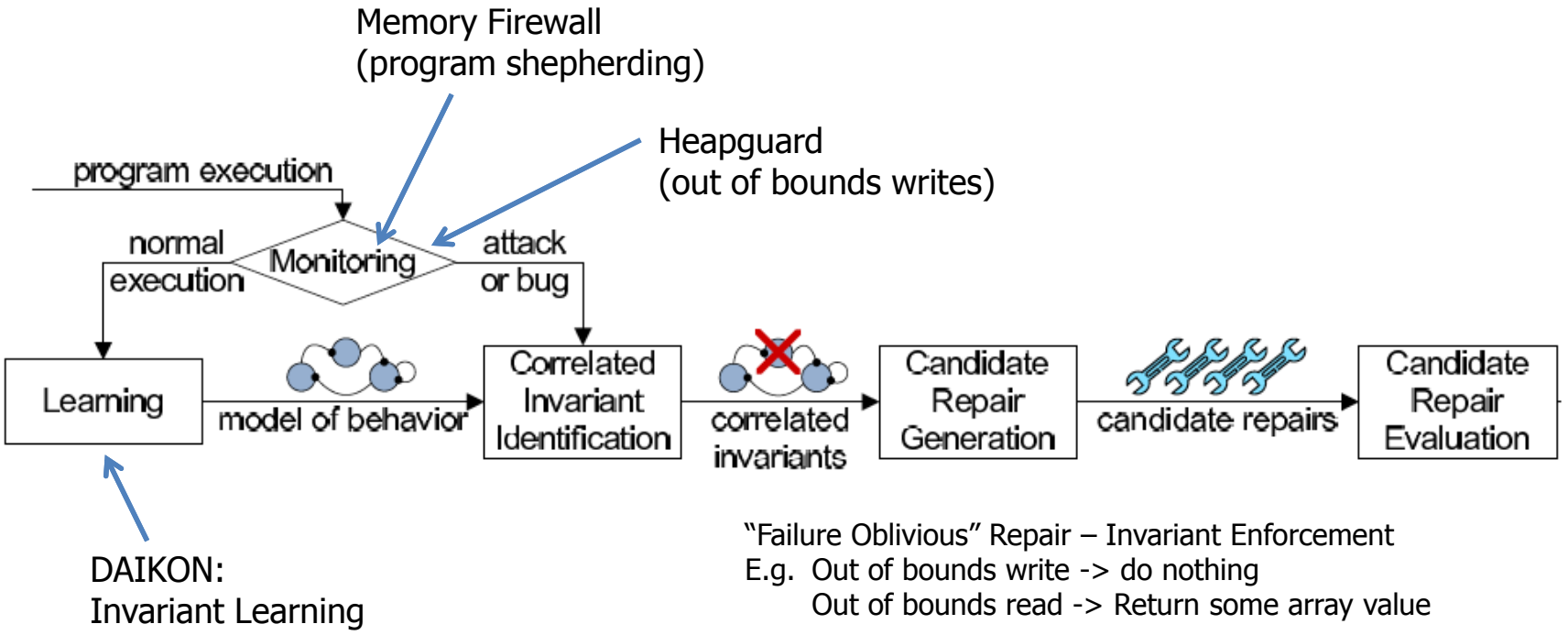
Adaptation selects the method which maximizes expected net benefit

AWDRAT Experimental Results

Attack Type	Number of Attempts	Detected	Diagnosed	Corrected
Hand Placed	**24**	**20**	**20**	**11**
Data MAF API	10	6	6	6
Data lower API	5	5	5	5
Port Opening	3	3	3	NA
File Write	3	3	3	NA
Application Launch	3	3	3	NA
Random	**26**	**25**	**25**	**9**
Data MAF API	6	5	5	1
Data lower API	8	8	8	8
Port Opening	5	5	5	NA
File Write	4	4	4	NA
Application Launch	3	3	3	NA
Wrapped	**9**	**9**	**9**	**9**
File Write	3	3	3	3
Port Opening	3	3	3	3
Application Launch	3	3	3	3

Diagnosis includes identification of where the failure occurred, the nature of the failure (bad event, corrupted data) and whether code-files vs. data files are likely candidates.

ClearView (SOSP 2009)



"Failure Oblivious" Repair – Invariant Enforcement
E.g. Out of bounds write -> do nothing
Out of bounds read -> Return some array value

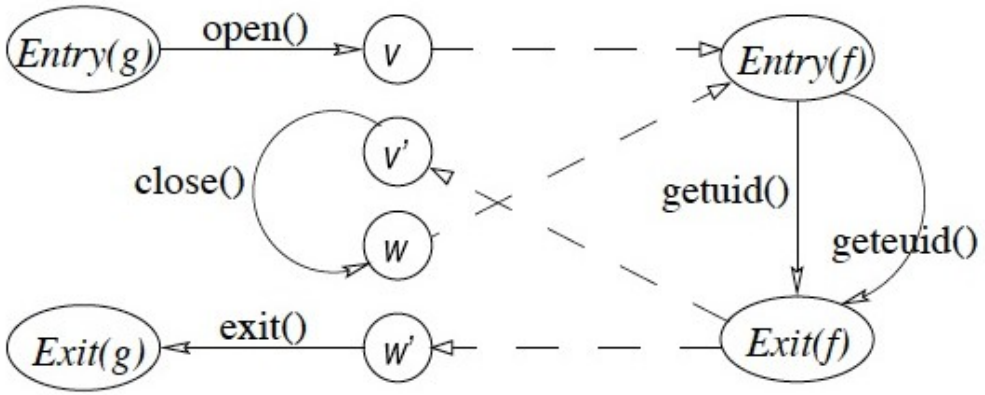
ClearView Configuration	Page Load Time (seconds)	Overhead Ratio
Bare Firefox	7.5	1.0
Memory Firewall	11.0	1.5
Memory Firewall + Shadow Stack	14.9	2.0
Memory Firewall + Heap Guard	19.0	2.5
Memory Firewall + Heap Guard + Shadow Stack	22.7	3.0

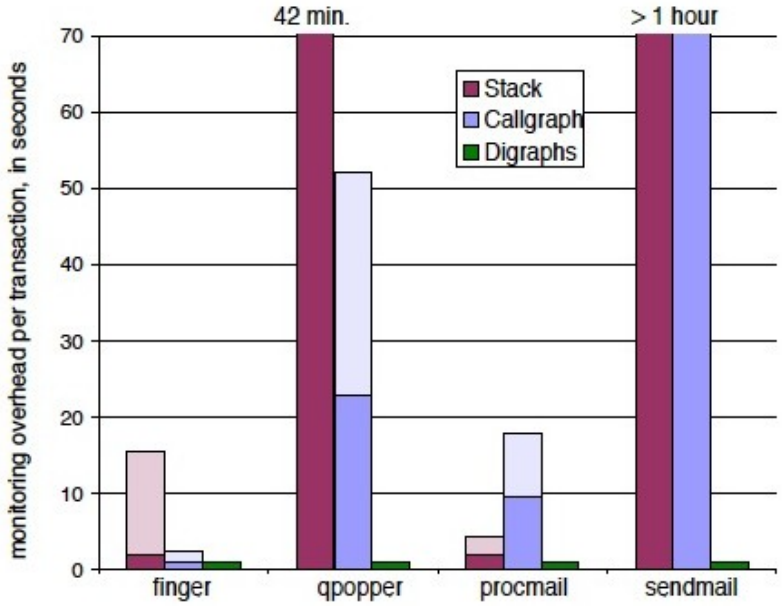
Bugzilla Number	Presentations	Error Type
269095	6	Memory Management
285595*	4	Heap Buffer Overflow
290162	4	Unchecked JavaScript Type
295854	5	Unchecked JavaScript Type
296134	4	Stack Overflow
311710	12	Out-of-Bounds Array Access
312278	4	Memory Management
320182	6	Memory Management
325403*	4	Heap Buffer Overflow

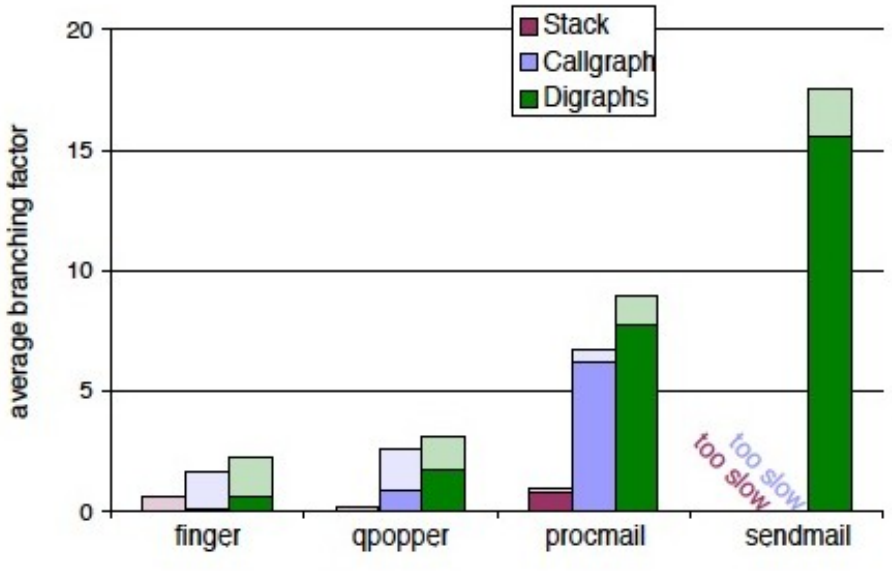
Static Analysis Builds Control Flow Graph

(Wagner & Dean, Oakland Security Conference 2001)

```
f(int x) {
  x ? getuid() : geteuid();
  x++;
}
g() {
  fd = open("foo", O_RDONLY);
  f(0); close(fd); f(1);
  exit(0);
}
```







Repair Using Evolutionary Computation
(Forrest ICSE 2009)

- How to repair a software fault in 5 (or 10) minutes or less using evolutionary computation
- Assume:
  - Access to C source code
  - Negative test case that executes the buggy code
  - Positive test cases to encode required program functionality
- Construct Abstract Syntax Tree (AST)
- Evolve repair that avoids negative test case and passes positive test case
- Minimize repair using program analysis methods

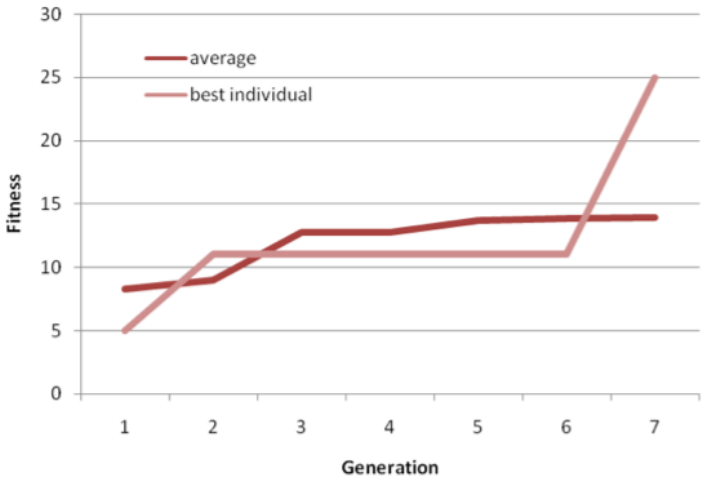
Example Repairs – Security Vulnerabilities

Program	Lines of Code	Path Length	Program Description	Vulnerability
nullhttp	5,575	768	webserver	remote heap overflow
openldap io.c	6,519	25	directory protocol	non-overflow denial-of-service
lighttpd fastcgi.c	13,984	136	webserver	remote heap overflow
atris	21,553	34	graphical game	buffer overflow
php string.c	26,044	52	scripting language	integer overflow
wu-ftpd	35,109	149	FTP server	format string
TOTAL	**108,784**	**1,164**		

Repairing the Zune bug using GP

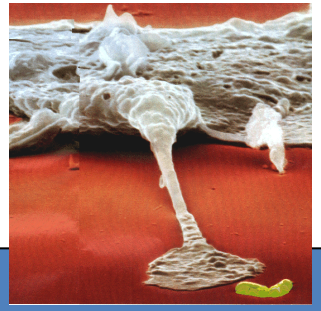
- Primary repair discovered in 37 seconds
- Final repair produced in 5 seconds
- To date have repaired 15/15 bugs in over 140,000 LOC

```
1   void zunebug_repair(int days) {
2     int year = 1980;
3     while (days > 365) {
4       if (isLeapYear(year)){
5         if (days > 366) {
6           // days -= 366; // repair deletes
7           year += 1;
8         }
9         else {
10        }
11        days -= 366;       // repair inserts
12      } else {
13        days -= 365;
14        year += 1;
15      }
16    }
17    printf("current year is %d\n", year);
18  }
```



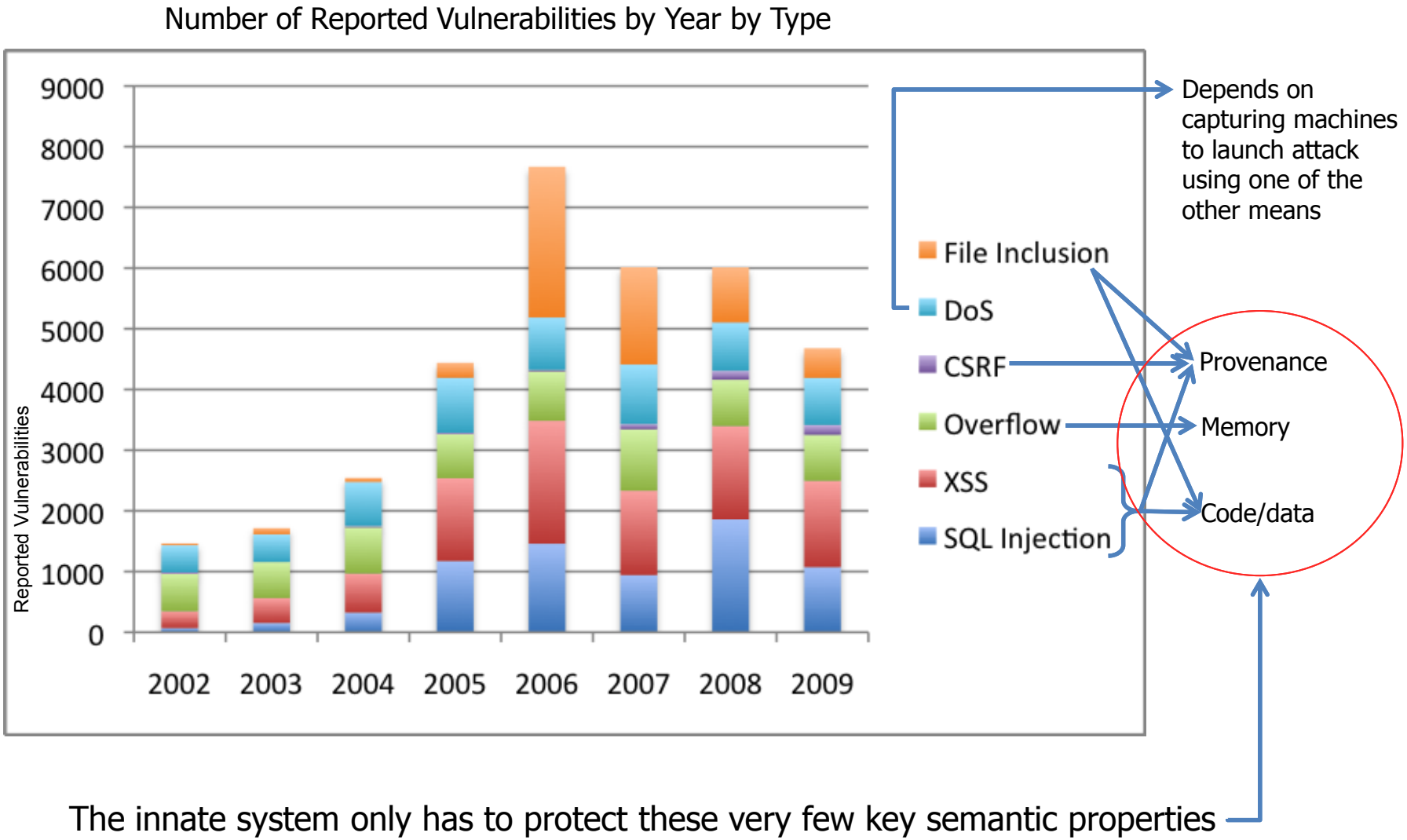
INNATE IMMUNITY: COMPLETE MEDIATION THROUGH HARDWARE ENFORCEMENT

Biology and Computation: Two Design Styles



Computation	Biological
Near Perfect Components	Fallible components
Core design formed in era of scarcity	Abundance of resources
Core design formed in isolated environment	Evolution in ecosystem of predators and parasites
Evolutionary pressure from market: price, performance and features	Evolutionary pressure from ecosystem: survivability
Self-regulation and adaptation rarely considered. Runs open-loop.	Self-regulation and adaptation are core mechanisms. Closed loop control.
No enterprise-wide survivability mechanisms	Diversity for population survival Public-health systems in human society

Almost All Current Technical Vulnerabilities Are Manifestations of a Few Root Causes



Meta-data Is the Key

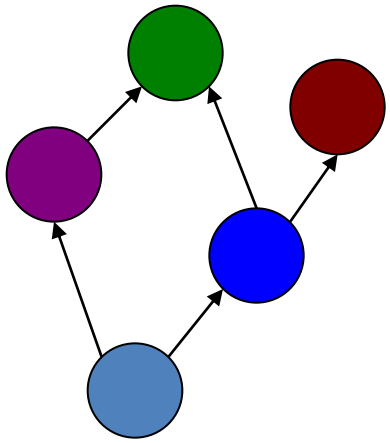
Compartments:

- Compartment = Collection of Data
- Organized in lattice

Principals:

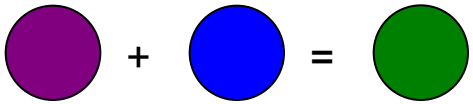
- Principal = An active entity
- Organized in lattice

- Everything, including Principals and Compartments, have representations as objects in a compartment.
- Each running process has a principal (who it's acting for) and a compartment (where it can allocate data)



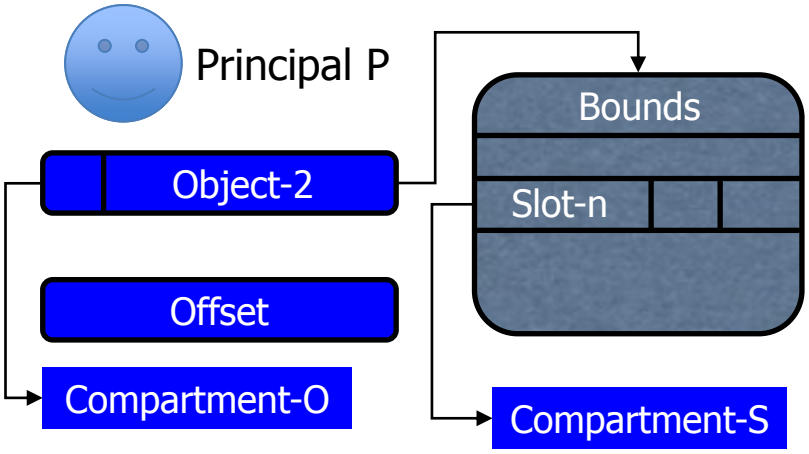
Access Rules:

- For each operation, a matrix of which classes of principals can perform the operation on data in which classes of compartments
- Specifies compartment of result
- Collectively forms a policy restricting flows between compartments

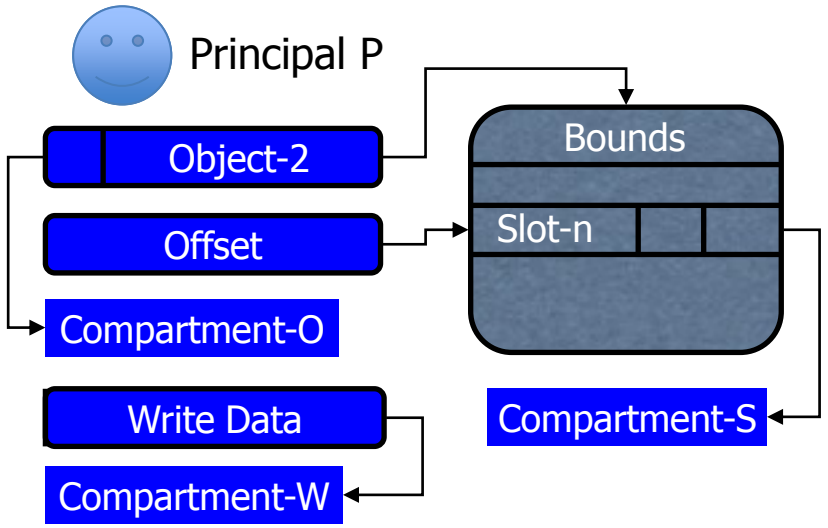


Rules: Data Structure Access

1. Can Principal P read from Compartment-O?
2. Can Principal P read slot-n of the object



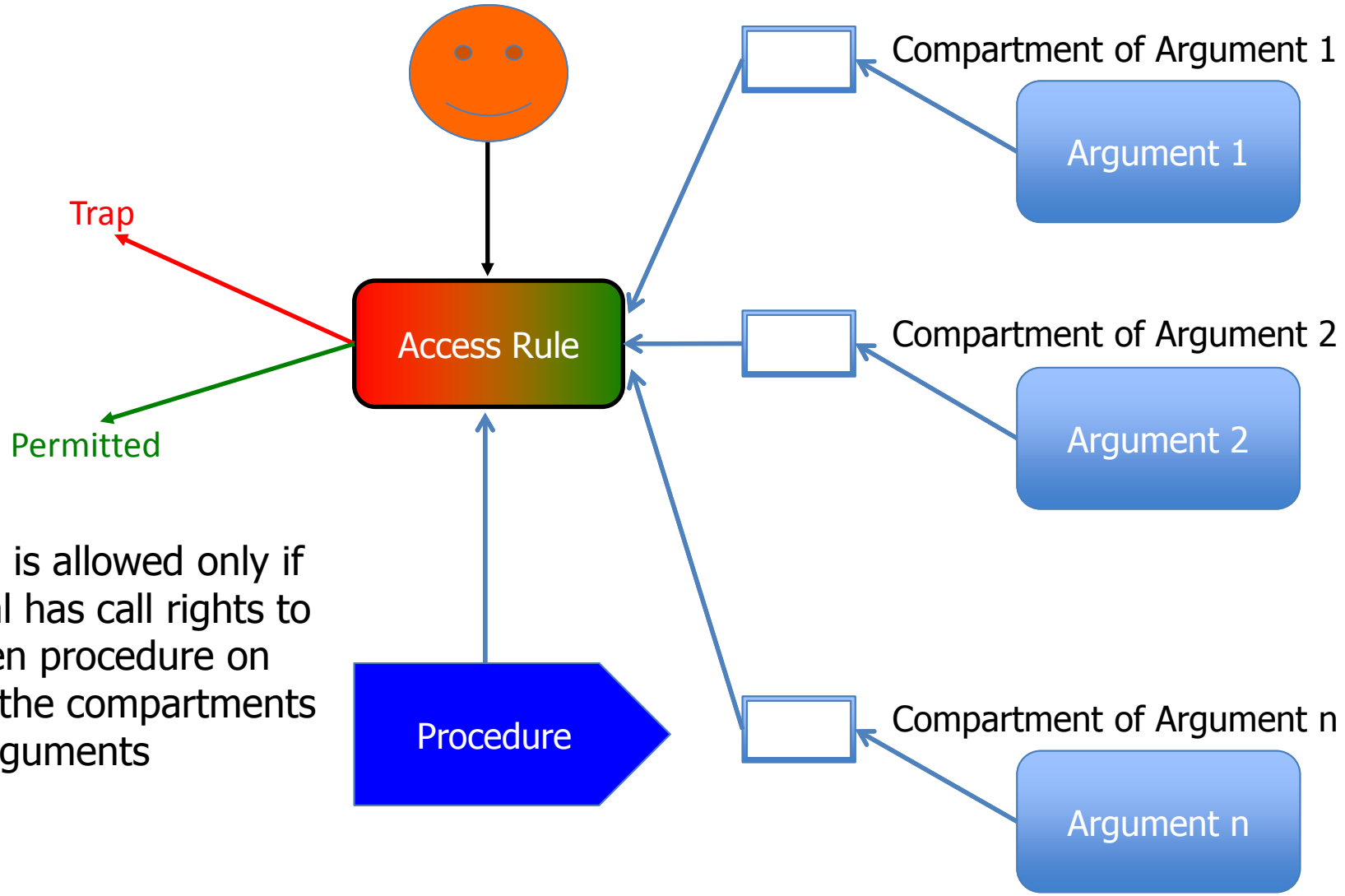
1. Can Principal P write to Compartment-O?
2. Can Principal P overwrite data in Compartment-S with data from Compartment-W?
3. Can P write to slot-n of the object?



Load

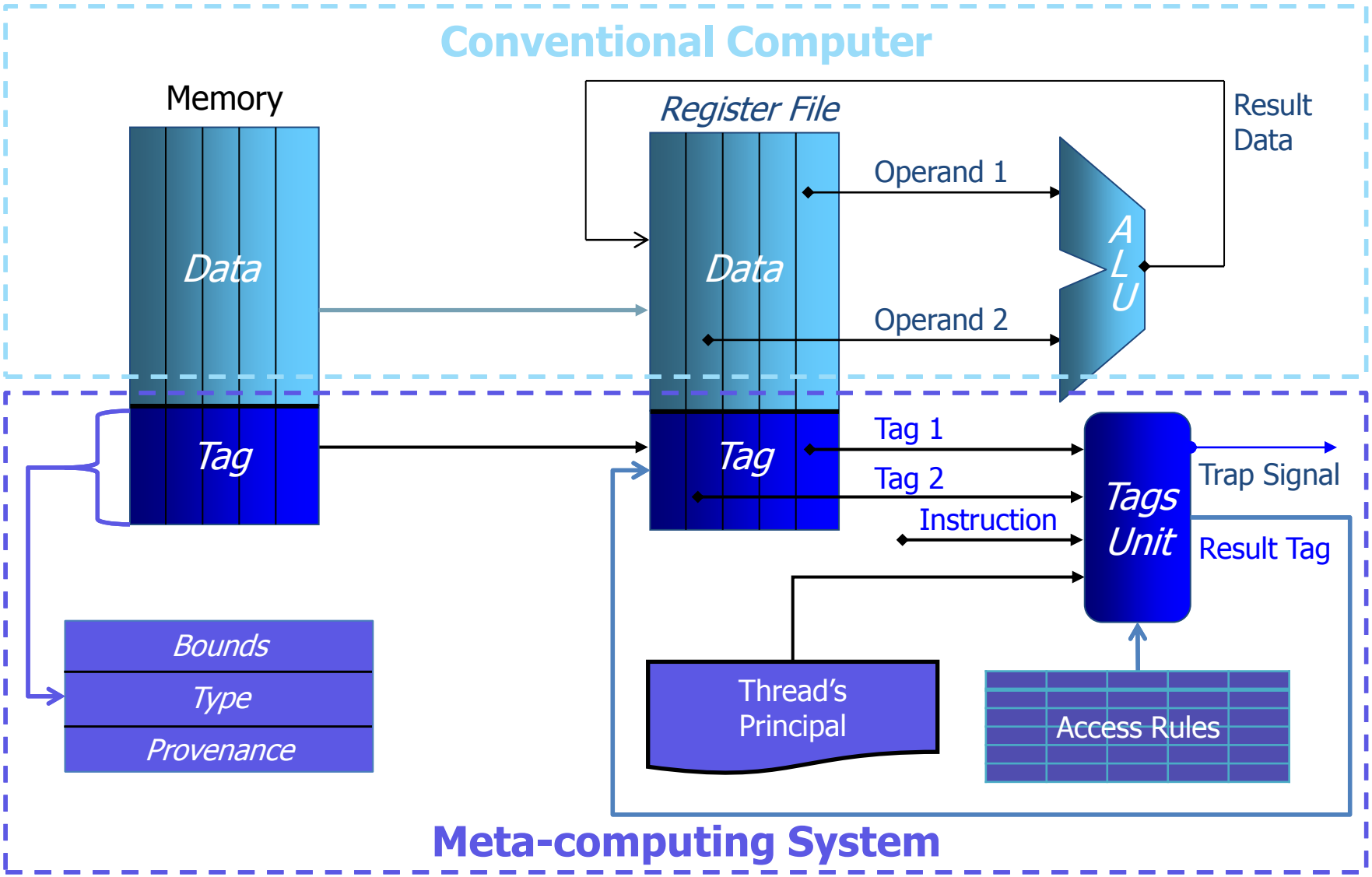
	C-1	C-2	C-3	C-4	C-5	C-n
Principal-1	Y	Y	trap	trap	trap	trap
Principal-2	trap	Y	Y	Y	trap	trap
Principal-3	Y	trap	trap	trap	Y	trap
Principal-n	trap	trap	Y	trap	trap	Y

Procedure Call Rules



The call is allowed only if Principal has call rights to the given procedure on data in the compartments of its arguments

Hardware Mediation

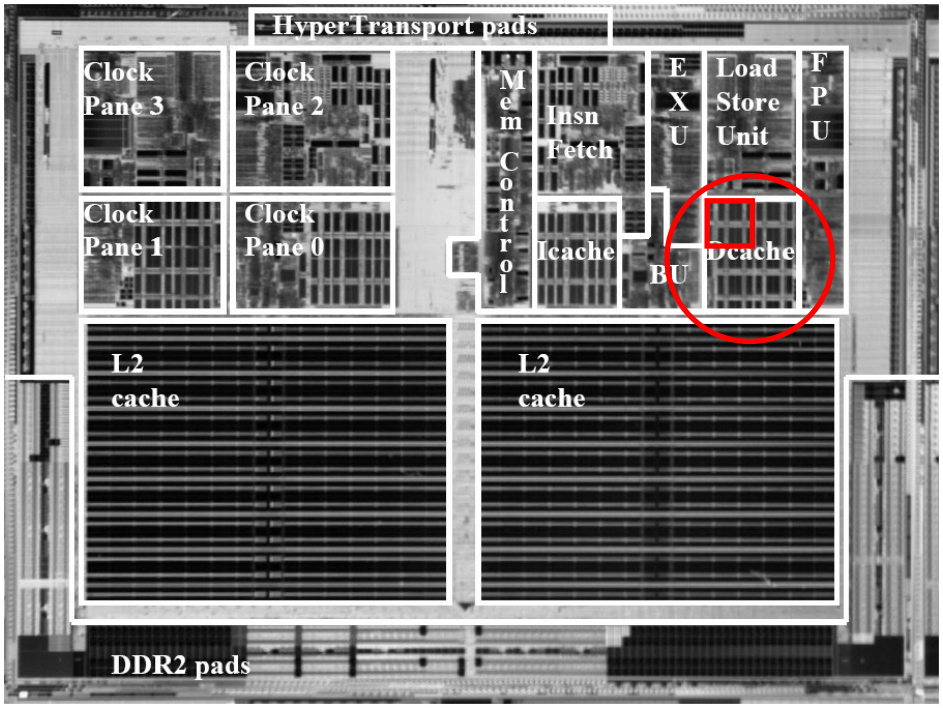


Perspective

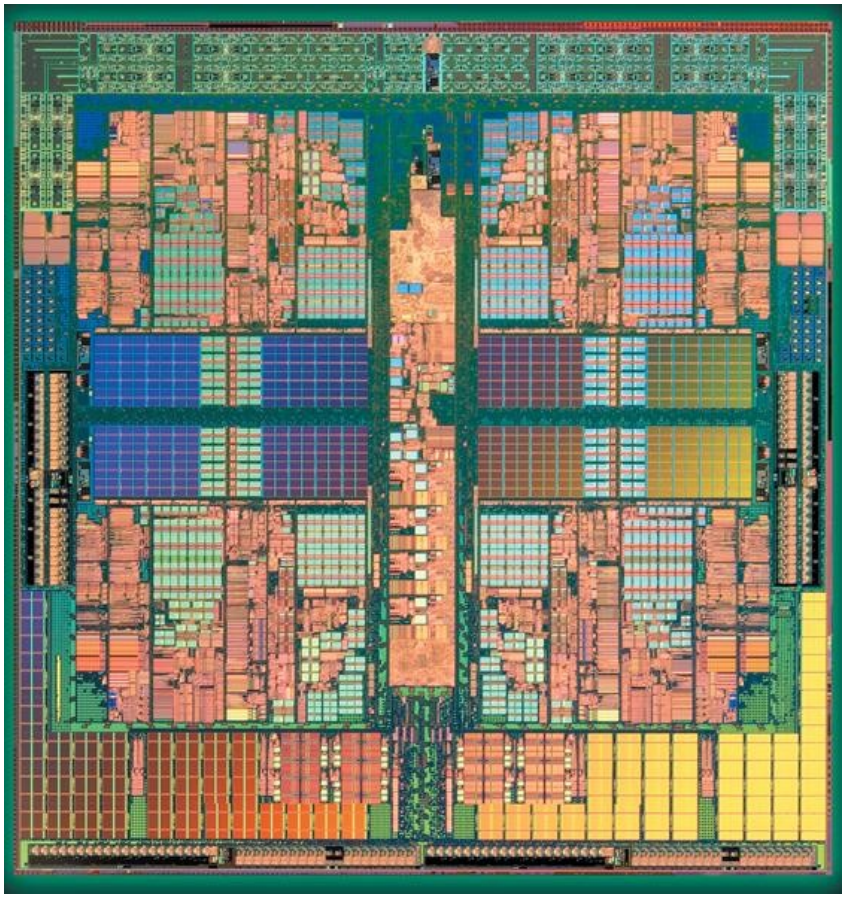
Tag Processing Unit is about 125K bits

125,000 SRAM bits < 16K Bytes

Note: L1 Dcache on Opteron is 64KB



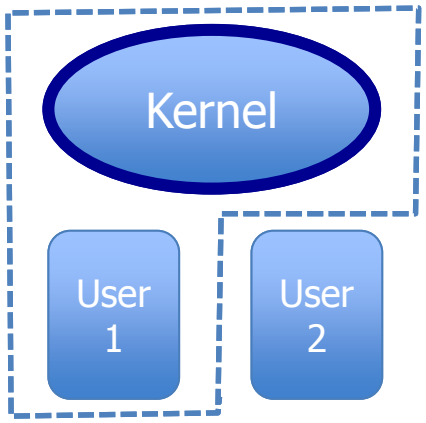
Dual and Quad Core Opterons



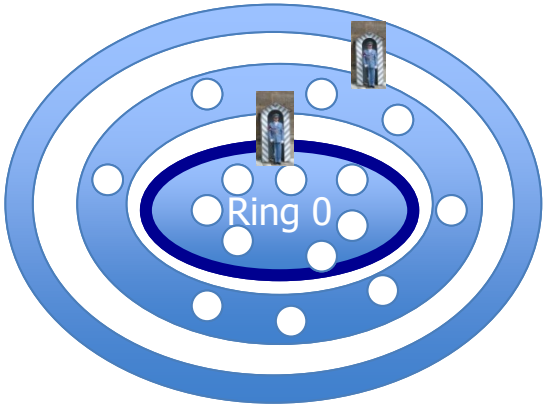
Computing in a Dangerous World

NEW SOFTWARE ARCHITECTURES ELIMINATING IMPLICIT TRUST

System Software Models



Kernel:
Windows, Unix

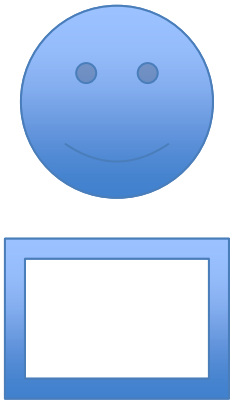


Rings: Multics

Components:
TIARA, Singularity
Separation Kernels

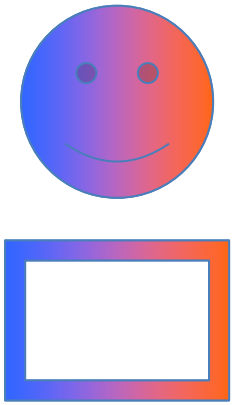
Separation via Compartments and Principals

- Each system component has its own **compartment** for its private data
- Each system component has its own **principal** for managing its private data
- System components may have "**satellite**" compartments and principals for controlled interactions with users and other system components
- Gates are used to control access to shared compartment and principals



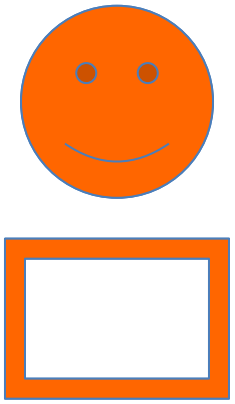
Process Manager

Gate to control interaction



Satellite for Interaction

Gate to control interaction



Log Manager

Gates

- The only way to change principal and compartment is by calling a "gate"
  - **Package of new principal, new compartment for allocating objects, and a procedure to be executed**
  - Rebinds the thread's principal and compartment during execution of its procedure
  - Each gate is itself an object and therefore lives in a compartment
- A class of access rules specifies which principals can invoke gates in which compartments

Analogy:

- She's not allowed to operate inside the castle
- He can go in and do something on her behalf
- Access rules dictate whether he'll answer her request
- He's not allowed to operate outside the castle
- While acting on her behalf he does not gain her privileges to areas outside the castle

Computing in a Dangerous World

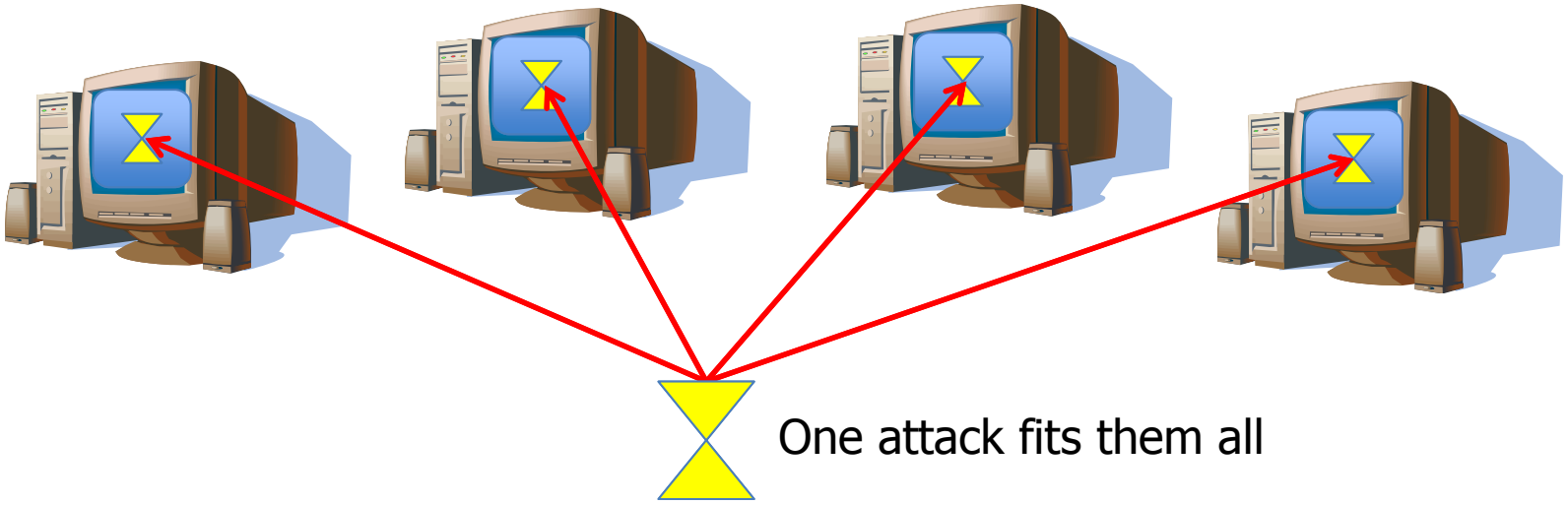
BEATING DOWN THE COMPUTATIONAL MONOCULTURE ADOPTING PUBLIC HEALTH MODELS

"Public Health" Infrastructure for Survivable Systems of Systems

- **Immune System** in one machine notices attack
- **Public Health System** distributes information
- **Quarantine** untrusted component systems
- Actively probe to **diagnose** attack
- **Immunize** the population
  - Public Health network distributes preventive measures
- Components **Evolve** by dynamically linking in fixes and preventive measures

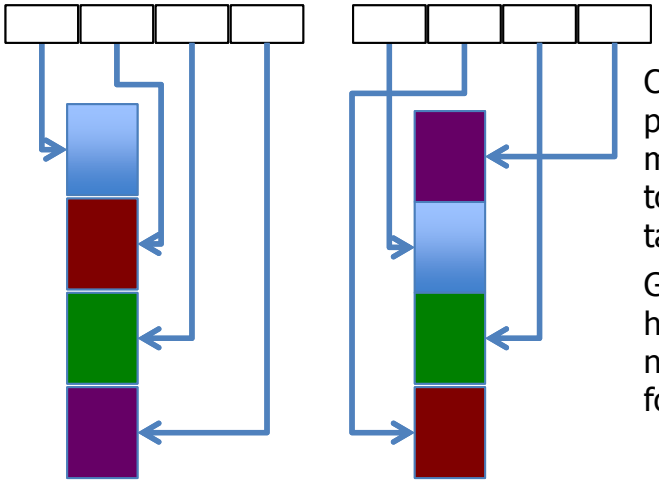
Monocultures are not survivable

- The attacker's work factor is proportional to Entropy
  - When all systems are the same, a single attack disables them all
  - When a single system never changes, the same attack will work repeatedly
- We currently have a computational monoculture.



Dynamic Diversity Makes a Single Host Different from Moment to Moment

Address space randomization



Code and/or data blocks are periodically repositioned in memory so that attacker has to work harder to find a target.

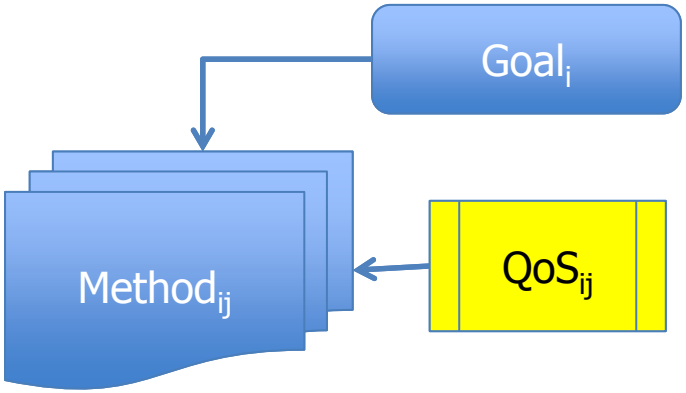
Garbage-Collected memory has the property inherently, new methods may optimize for increased entropy.

Instruction set randomization

Disk	Memory	ICache
Instruction-1	Encrypted-1	instruction-1
Instruction-2	Encrypted-2	instruction-2
Instruction-3	Encrypted-3	instruction-3
Instruction-4	Encrypted-4	instruction-4
Instruction-5	Injected-1	Encrypted-1
Instrution-6	Injected-2	Encrypted-1
	Encrypted-5	instruction-5
	Encrypted-6	instruction-6

Functional Redundancy & Decision Theoretic Dispatch

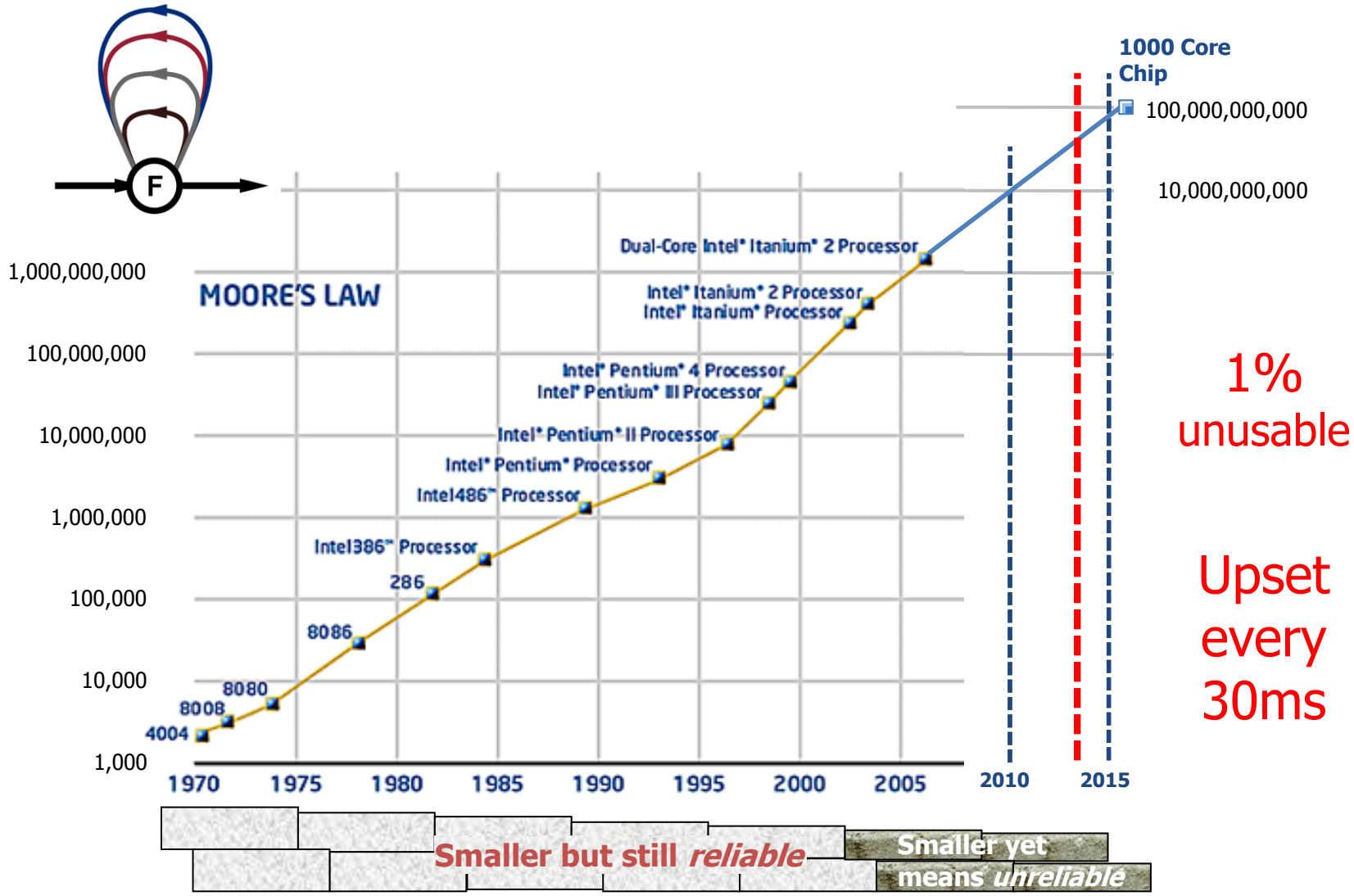
There are multiple methods for achieving each goal ("n-version programming"). Each distinct method has different qualities of service. Method selection is driven both by preferences over QoS and by need for unpredictability.



Code is encrypted as it enters memory and Decrypted as it enters the instruction cache (or translation buffer). Injected code in native instruction set is then encrypted and not executable. Encryption key can be varied by process and time.

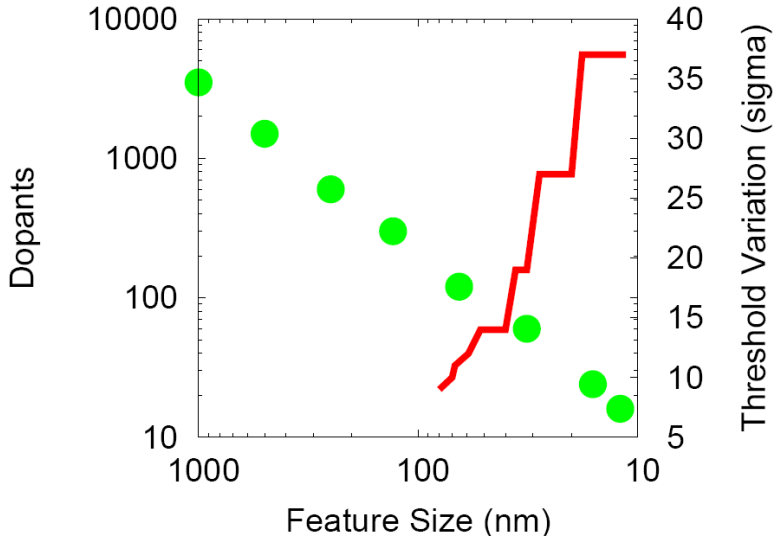
YOU'RE NOT PARANOID
MOTHER NATURE IS OUT TO GET YOU

DARPA Challenge

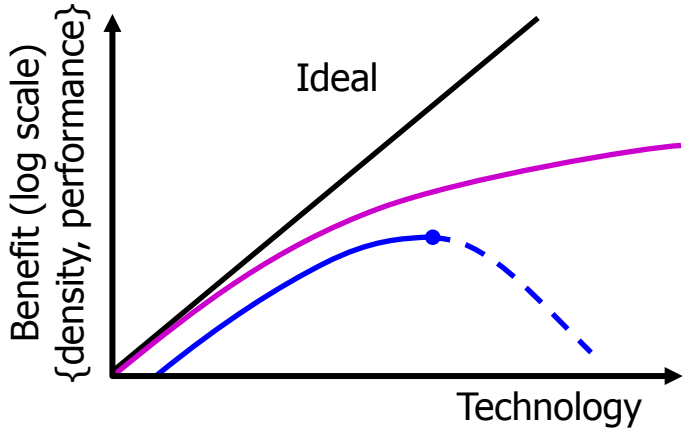


Premature Death of Scaling

- Must scale voltage ↓ for energy and power density
- Near atomic-scale effects demand ↑ voltage margins
  - Discrete dopants and atomic size
  - Upsets
  - Aging

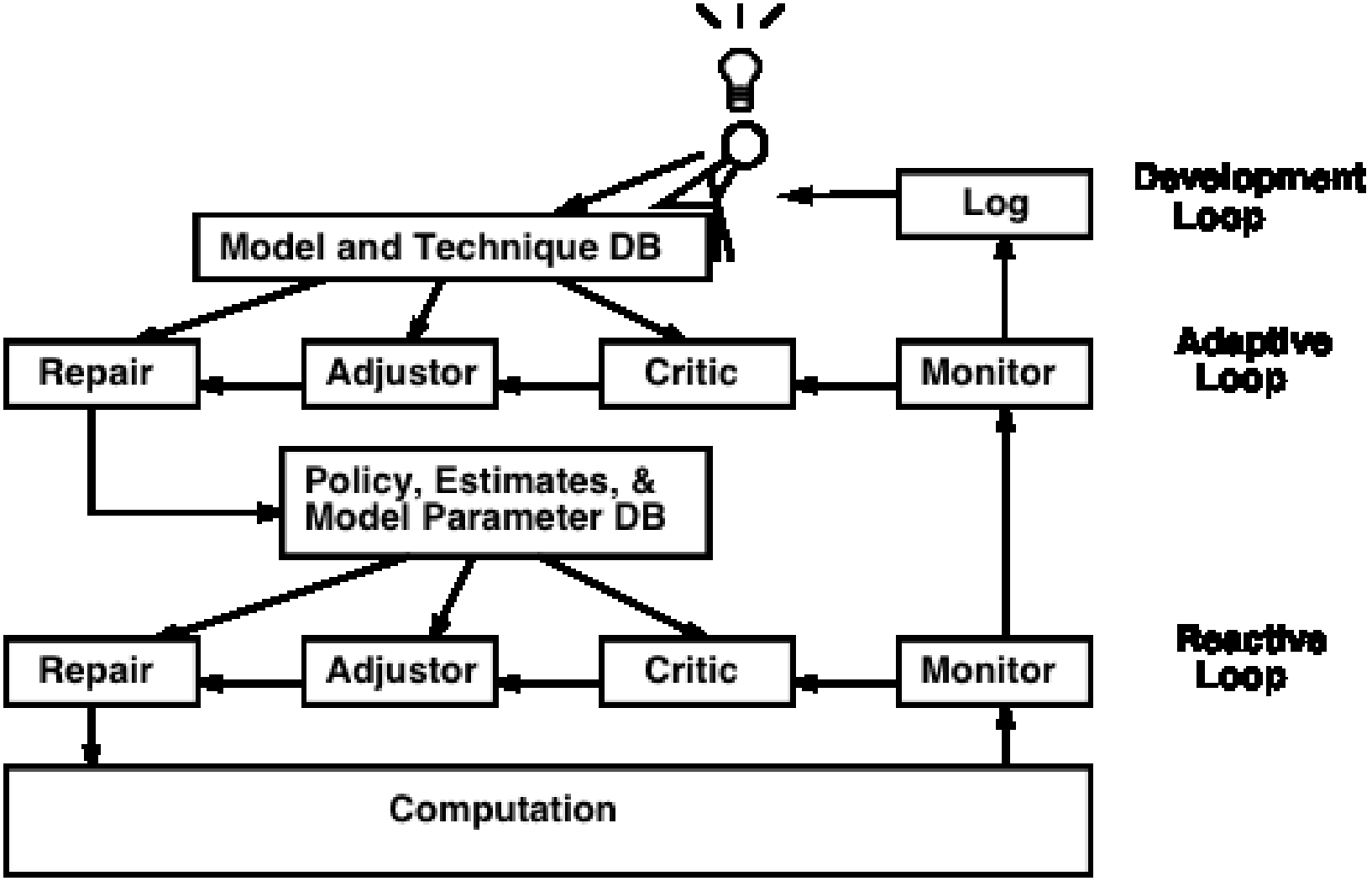


Uncertainty and reliability issues impact all future technologies.



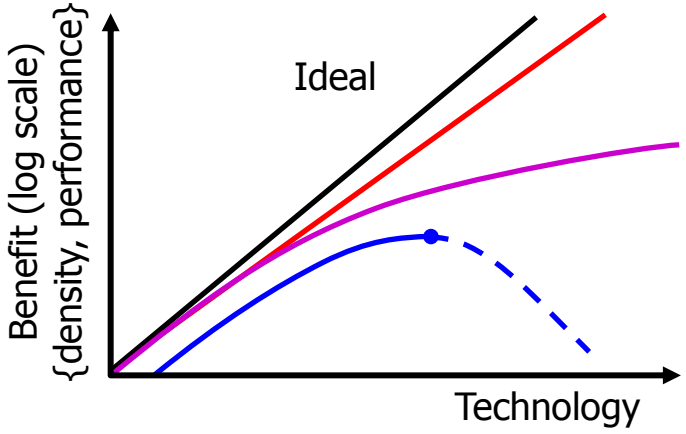
DeHon, Knight, Savage, Shrobe, Smith — June 2009

DAMS – Multiple Levels of Self Monitoring and Adaptation



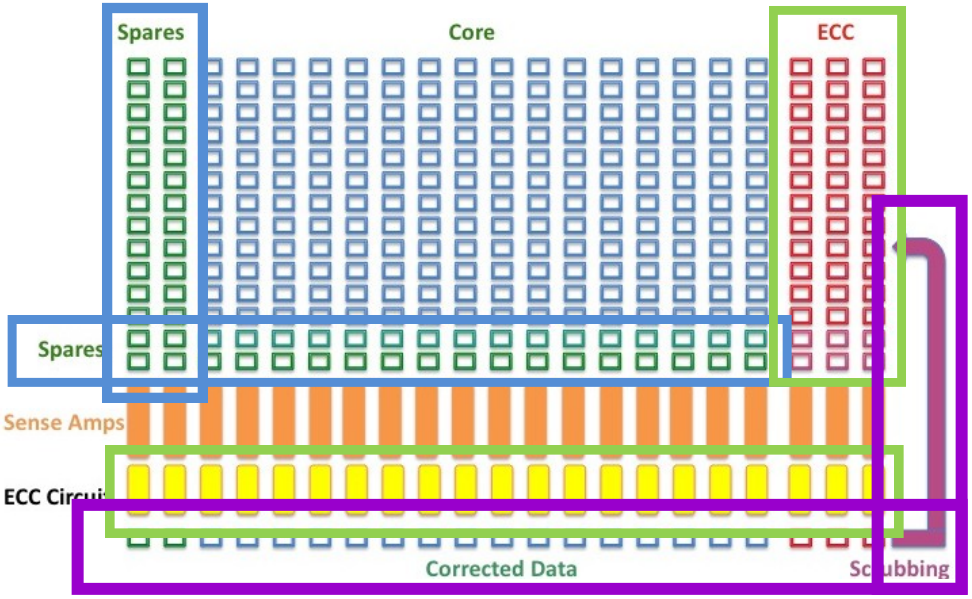
Challenge Problem

- Chip: 103 cores (10^{11} transistors)
- Threshold voltage variation ~27% (ITRS 22nm)
  - 1% gates unusable initially at target energy
- Frequent upsets – every 30 milliseconds
  - Transient flips, permanent failures, aging slowdown
  - 1 Billion transistors/year fail (1% capacity)
- What is a workable adaptation and repair strategy?
  - Does it suffice to discard a core on hard error?



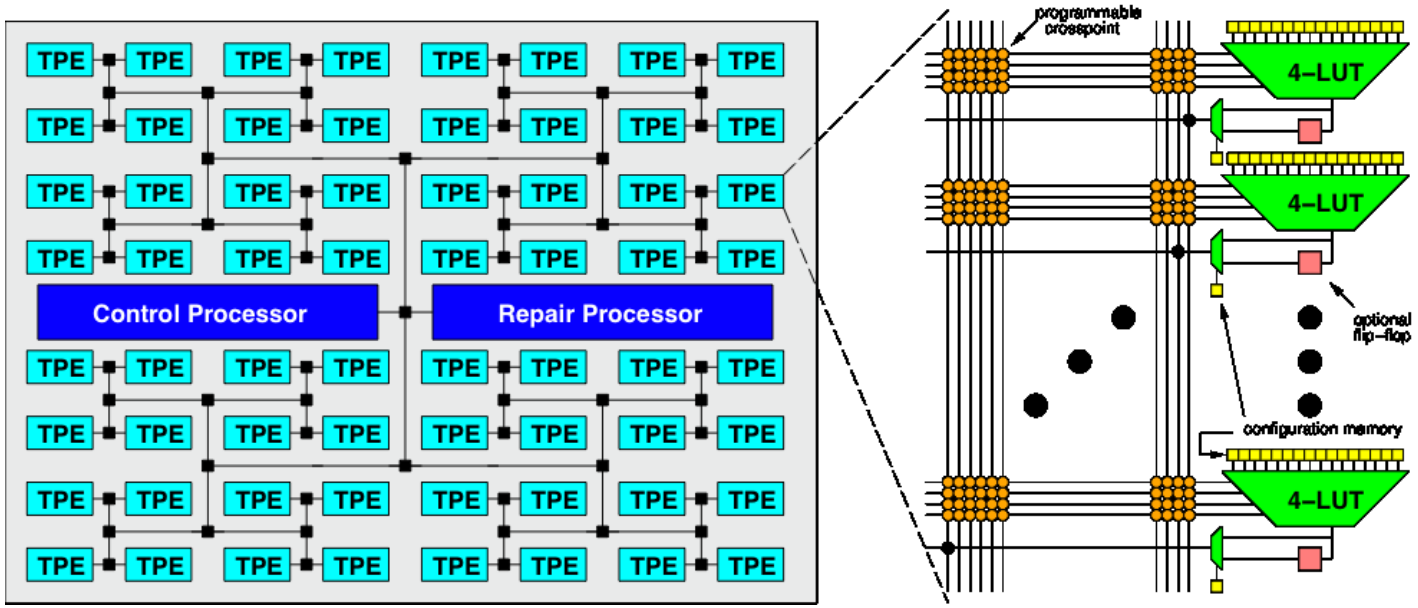
Inspiration from Memory Systems

- Correction strategy
  - Row/column sparing
  - Error-correcting codes
  - Scrubbing
- Differential reliability
  - Judicious mix of large and reliable vs. small but unreliable gates
- Achieves roughly the size and energy savings of small cells with the reliability of large ones



DeHon, Knight, Savage, Shrobe, Smith — June 2009

Tiled Chips



980 Task Processing Elements (TPE) each comparable to Pentium IV 11nm geometry

Control and Repair processor 140nm geometry are highly reliable

200 ms computational chunks farmed out to TPEs

Validity check and invariant check performed during and at end of computation chunk

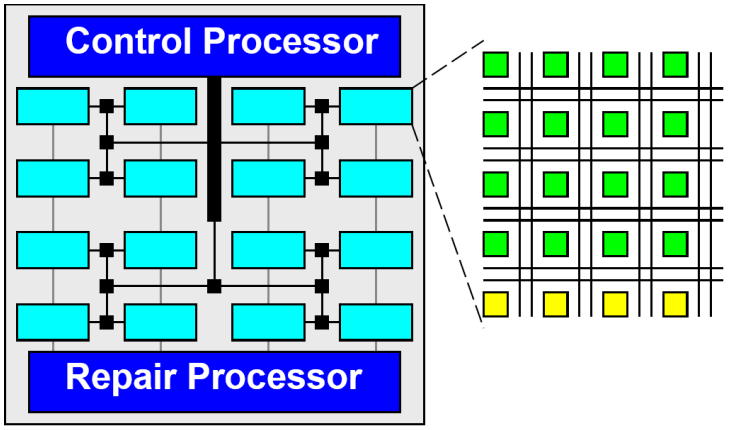
About 7% failure rate due to transients, retried under control processor supervision

If we discard a core in response to a hard error, then all TPEs have persistent fault after 5 minutes and whole chip is dead

But, if TPEs are made from reconfigurable logic can keep chip healthy for 10 years

An Example

- **Matrix Solve:** Ax=b
- **Check:** Compute residue |Ax-b|
- Transient error in arithmetic
  - May not matter → won't notice
  - Convergent algorithm → may just slow down
  - Corrupt result → detect with check, recompute
- Hard error in arithmetic
  - Re-execution → also fails check
  - Diagnose core with self-test
  - Reconfigure to repair
- Higher-level loop monitors policy effectiveness



Computing in a Dangerous World

It **is** dangerous out there

but we **can** adapt and we'll survive