Breeze: A Language For Writing Secure Software

Cătălin Hriţcu

SAFE

Northeastern

BAE SYSTEMS

The SAFE team

May 2011 @ San Jose: Sumit Ray, Howard Reubenstein, Andrew Sutherland, Tom Knight, Olin Shivers, Benjamin Pierce, Ben Karel, Benoit Montagu, Jonathan Smith, Catalin Hritcu, Randy Pollack, André DeHon, Gregory Malecha, Basil Krikeles, Greg Sullivan, Greg Frazier, Tim Anderson, Bryan Loyall.

Missing or added since May 2011: Greg Morrisett, Peter Trei, David Wittenberg, Amanda Strnad, Justin Slepak, David Darais, Robin Morisset, Chris White, Anna Gommerstadt, Marty Fahey, Tom Hawkins, Karl Fischer, Hillary Holloway, Andrew Kaluzniacki.

Sony Makes it Official: PlayStation Network Hacked

PlayStation Network was taken offline three days ago, all eyes fell on the Anonymous

SEPTEMBER 07, 2011

Debacle deepens for hacked SSL certificates issuer

Report indicates widespread compromise of DigiNotar's infrastructure questions industry's response to breaches

By Robert Lemos | InfoWorld

Print

The attack on DigiNotar, an issuer of

certificates, appears more

thought. A report by the

the breach

fraudulen

substantial

infrastructure

HTC admits to 'serious' security flaw on smartphones

October 5, 2011

Smartphone maker HTC has admit

smartphones being vulne

to find out where you are,

messages to and disable

On Tuesday Fairfax Media

Android smartphones that

app that is granted access to

require. Once granted access

access to data that was shown

Predator Drones: Has Virus Invaded Potent U.S. Weapon?

Article | Comments

Tweet

Email | Order Reprints

Text Size

the Predator and Reaper drones, may have b

which alleged al-Qaida terrorist Anwar al-Awlak

U.S. Air Force declined comment on the report

ated by Wired.

keep wiping it off and it keeps coming back,"

official said to the publication.

the Pentagon would unlikely confirm its has

ted to yet another cyber-attack, the virus

ger one of the most deadly and popular mi

which have proved especially effective in

stan, as well as other venues, like Yemen

The Stuxnet outbreak

A worm in the centrifuge

An unusually sophisticated cyber-weapon is mysterious but important

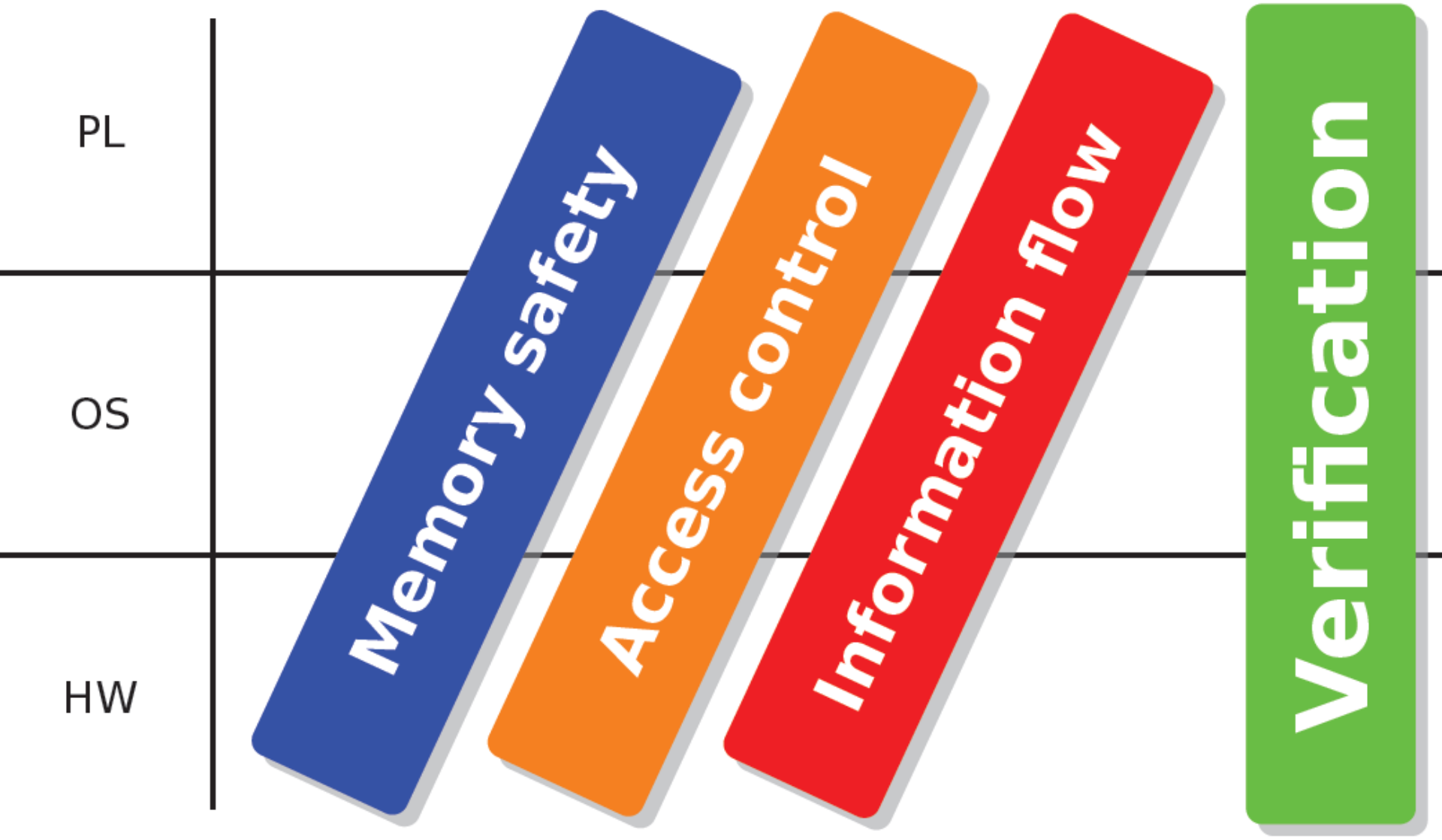
Sep 30th 2010 | from the print edition

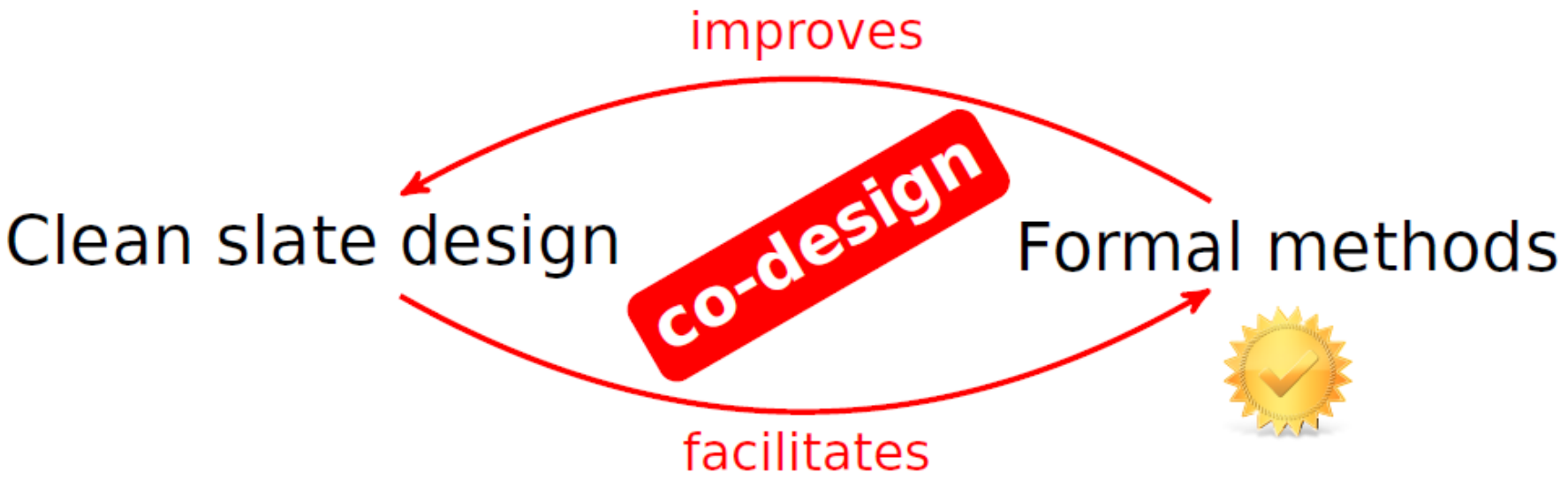
Common Weaknesses Enumeration: Top 25

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

http://cwe.mitre.org/top25/index.html#Listing

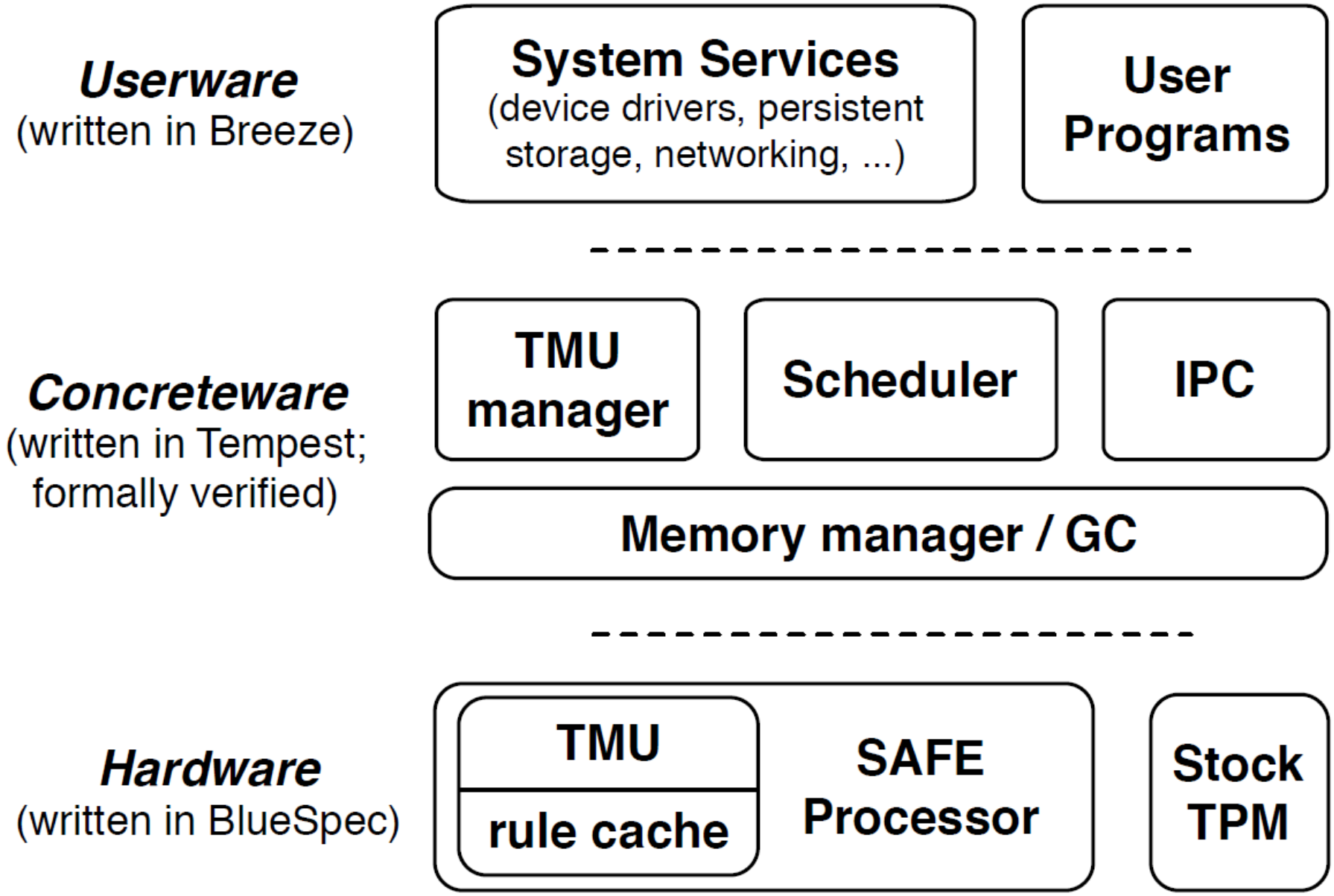
The SAFE project





Occasion to try new abstractions
Simpler, cleaner design
Fully correct wrt. specs

Precise specifications
Global guarantees
Machine checked proofs

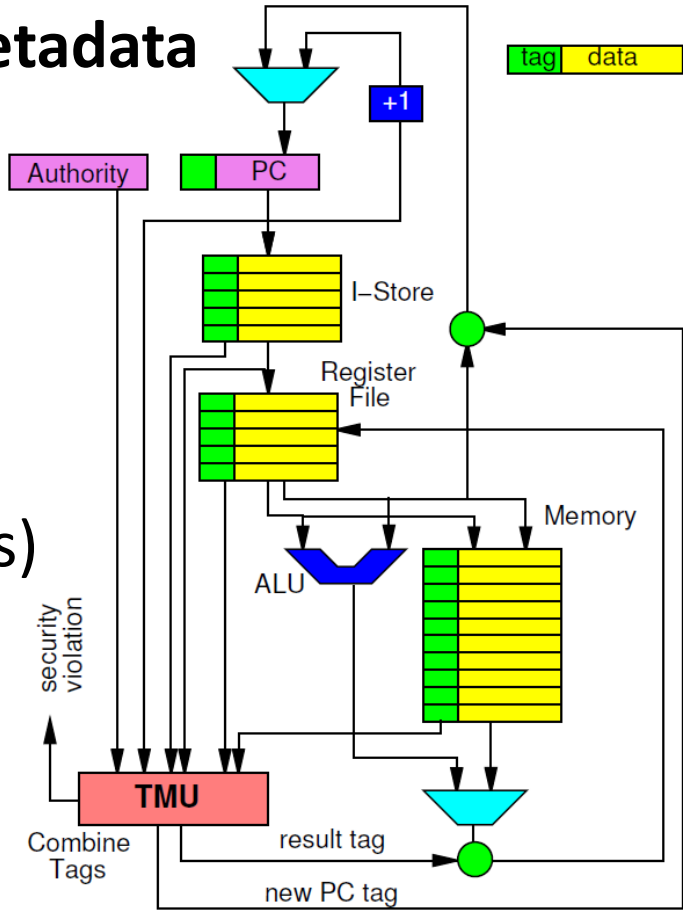

Userware
(written in Breeze)
System Services
(device drivers, persistent storage, networking, ...)
User Programs
Concreteware
(written in Tempest; formally verified)
TMU manager
Scheduler
IPC
Memory manager / GC
Hardware
(written in BlueSpec)
TMU
rule cache
SAFE Processor
Stock TPM

Breeze

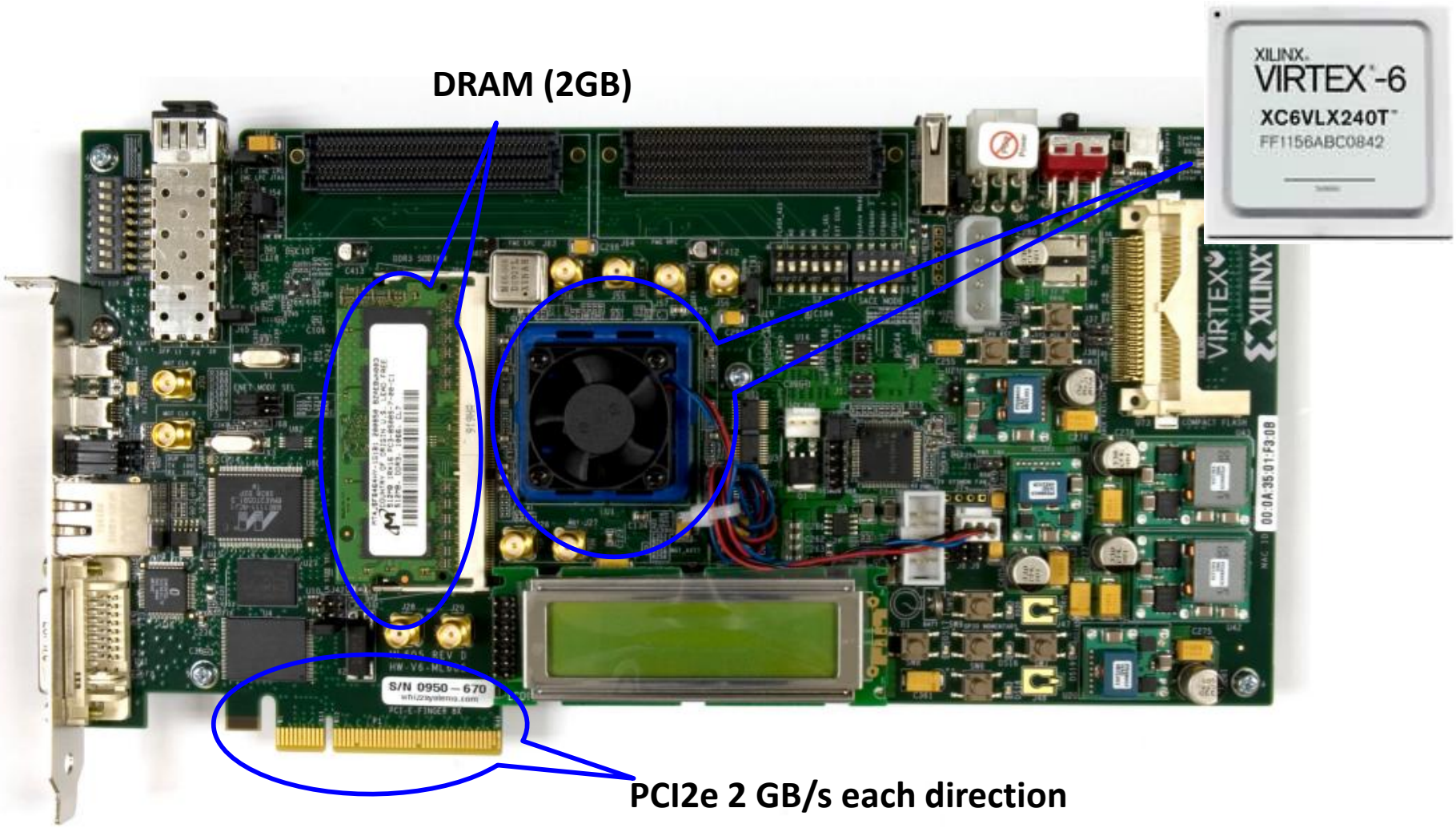
- High-level PL for writing **secure** software
  - user programs & system services (+ security policies)
- **functional core** (λ) + concurrency (π) + state (!) …
- **dynamically typed** (for now)
  - easier to experiment with
  - directly reflects capabilities of HW
  - dynamically-checked first-class **contracts**
- **fine-grained dynamic information flow control**
- **access control** (more later on why we need both)

Fine-grained dynamic IFC

- Every Breeze value has an IFC label: $\langle \rho, \lambda x.x \rangle @ l$
- At the SAFE HW level, **every word is tagged**
  - tags are pointers to **arbitrary metadata**
  - **checking happens in parallel**
    (HW rule cache speeds this up)
  - TMU is **programmable**
    (rules defined in software)
  - **composable** rule models
    (IFC labels = one of many models)
  - primitive TMU rule models:
    - can't add two pointers,
    - can't execute an integer, etc.
    - only scheduler can create threads



SAFE hardware prototype



IFC Labels in Breeze

- Started with Decentralized Label Model (DLM) by Liskov & Meyers (A → [B, C], D → [C,D])
- Later realized **DLM is not the only option**
  - Asbestos, HiStar/DStar, FLUME, HAILS, …
    all have their own decentralized … label models
- Breeze has "generalized label model"
  - parameterized by join, meet, principals, authority, etc.
  - hope to also support HAILS DC labels in the future
- How labels look like = the boring part of the story
  - in the **dynamic IFC** setting:
    the interesting part is how labels behave at runtime

Dynamic IFC: is it even possible?

- Non-interference is not a property (2-hyperproperty)
- Until around 2007 **folklore in PL community**: **non-interference can't be enforced dynamically** because of implicit flows
  - at least not without multi-execution or static information about the branches that are not executed
- Dynamic techniques pioneered IFC [Fenton, CJ '74]
- The lack of non-interference proofs didn't stop the OS community from building dynamic IFC systems (Asbestos, HiStar/DStar, FLUME …)

Purely dynamic IFC: Yes We Can!

- "From dynamic to static and back: Riding the roller coaster of IFC research" [Sabelfeld & Russo, PSI 2009]
- purely dynamic analysis (monitor) for termination-insensitive non-interference
  - termination-sensitivity hard to achieve in any way
- "flow insensitive" analysis
  - labels of mutable variables can't change
  - precise flow sensitive dynamic analysis proved impossible [Sabelfeld&Russo, CSF 2010]
  - flow insensitivity is not such a big deal for a new language
    - ML references / Java variables + fields only have weak updates
    - Breeze channels have label for contents fixed at creation time

Purely-dynamic IFC: functional setting

- "Efficient purely-dynamic information flow analysis" [Austin & Flanagan, PLAS 2009]

$$\frac{}{\rho, pc \vdash \lambda x.e \Downarrow \langle \rho, \lambda x.e \rangle @ pc} \qquad \frac{\rho(x) = v@l}{\rho, pc \vdash x \Downarrow v@(l \vee pc)}$$

$$\frac{\begin{array}{l}\rho, pc \vdash e_1 \Downarrow \langle \rho', \lambda x.e \rangle @ l_1 \\ \rho, pc \vdash e_2 \Downarrow v_2 @ l_2 \\ \rho'[x \mapsto v_2 @ l_2], pc \vee l_1 \vdash e \Downarrow v_3 @ l_3\end{array}}{\rho, pc \vdash e_1\ e_2 \Downarrow v_3 @ l_3}$$

- Termination-insensitive non-interference

$$\left.\begin{array}{l}\rho_1, pc \vdash e \Downarrow v_1 @ l_1 \\ \rho_2, pc \vdash e \Downarrow v_2 @ l_2 \\ \rho_1 \simeq_l \rho_2\end{array}\right\} \Rightarrow v_1 @ l_1 \simeq_l v_2 @ l_2$$

The “infectious pc” problem

- pc automatically “tracks” branch labels
  - in order to prevent implicit flows
  - `copy = (if secret@H then true@L else false@L)`
- pc “infects” all values created on high branch
- this leads to **deeply high-labeled values**:
  - `if true@H then [1,2] else [] ==>`
    `(cons (1@H) (cons (2@H) (nil@H))@H)@H`
- we needed “infectious pc” because of **automatic pc lowering** on control flow merge points

First attempt to “manual pc lowering”

- Using pc declassification, really bad idea (adds tons of completely spurious audit points)

Catalin Hritcu <catalin.hritcu@gmail.com>

[Safe-breeze] Manual PC declassification considered harmful

6 messages

Benjamin C. Pierce <bcpierce@cis.upenn.edu> **Wed, Aug 17, 2011 at 1:24 PM**
To: safe-breeze@lists.crash-safe.org

A few months ago, we made the decision that it was better to remove the "automatic declassification of the PC" at the ends of conditionals and functions in Breeze (and at return instructions in the ISA) and, instead, demand that programmers lower the PC manually, if it becomes higher than they want it. Over the past few days, we've finally made this change to Breeze and have been experimenting with programming in this style. Our conclusion, sadly, is that it doesn't work.

The “poison pill” problem

- Fine-grained, dynamic IFC with decentralized LM
- Any code can classify data
  - `(P,_,_) = newPrin “P”; pill = 42@(P -> [P])`
- High data can be hidden under low labels
  - `x = [1,2,pill]@L; send cpub x`
- IFC violations are dynamic errors
  - `recv cpub x; map ((+) 1) x`
- Labels themselves are an IF channel
  - `x = (if secret@H then ()@H else ()@top); copy = (labelOf x == H)`

Non solution(s)

- Labels are an IF channel, so **hide labels**

$$\frac{\rho, pc \vdash e \Downarrow v@l}{\rho, pc \vdash \mathsf{labelOf}\ e \Downarrow l@l}$$

- Threads get killed on IFC errors
  - critical components (scheduler, allocator, drivers, …) get killed on reading poison pill
- Make channels unreliable?
  - contract on channel, silently discards poison pills
  - restore reliability using threads and timeouts?
  - so we can still write labelOf using a timing channel

Brackets: killing two birds with one stone

- Brackets = construct for **manual pc lowering / restoring**
  - automatic pc lowering played big role in creating our 2 problems
- Programmer has to **predict pc + label on all c.f. branches**

$$\frac{\rho \vdash e, pc \Downarrow v@l, pc' \quad l \sqsubseteq lb \quad pc' \sqsubseteq lb}{\rho \vdash lb\,\langle e \rangle, pc \Downarrow v@lb, pc}$$

- Brackets are **not a declassification construct**
  - always safe, unlike manual pc declassification
- Final label cannot depend on secrets
  - `top<if secret@H then ()@H else ()@top> ==> ()@top`
- Labels are now public, **no more "poison pills"**
- **No more "infectious pc"**
  - `H<if true@H then [1,2] else []> ==> [1,2]@H`

Attacker model

- Why does Breeze also have access control?
  - Isn't IFC enough?
- **IFC doesn't protect against malicious code!**
  - there will always be covert channels
  - malicious code can extremely easily exfiltrate secrets
  - I can write exfiltrator in 5 minutes, without timer
- Big part of IFC community seems to ignore the problem and accept **bogus attacker model**
  - We won't!

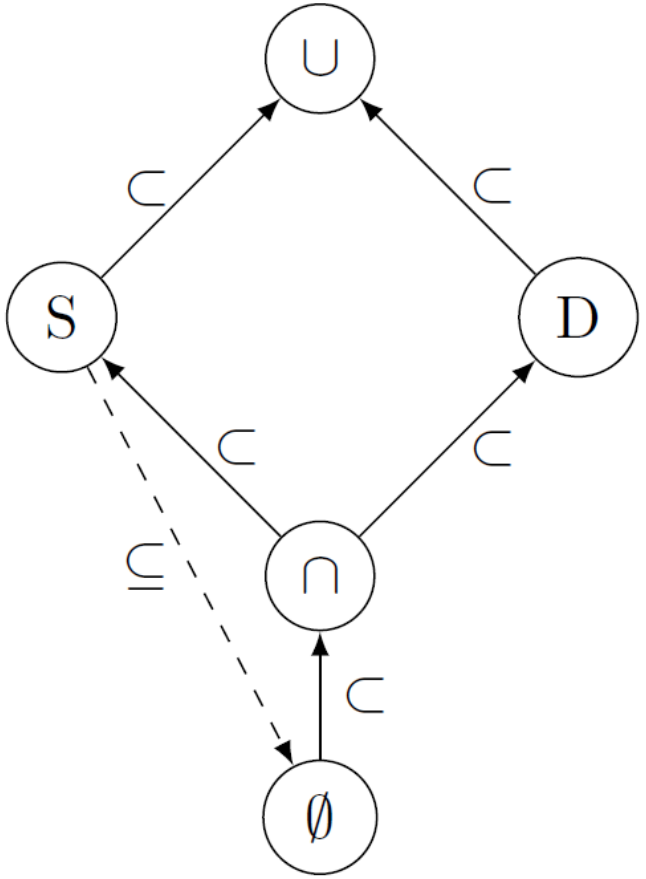
Authorities in Breeze

- Unprivileged Breeze code **cannot**
  - read and compute with secrets (and thus leak them over covert channels)
  - declassify secrets / endorse tainted values

  } separate capabilities
- Authorities are first-class values = **capabilities**
  - (P,Pe,Pd) = newPrin “P”; setAuth Pe in …
- Ambient authority makes this workable
  - otherwise pass authority on each little ISA instruction?
  - authority can also be attenuated (least privilege)
- Access control decision based on:
  - ambient read authority + IFC label of the data

Ambient authority propagation strategies

- Q: Do closures capture creation-time authority? (S = lexical propagation)
- Q: Is authority passed from caller to callee? (D = dynamic propagation)
- Capable experiment by G. Mallecha & G. Morrisett
- Breeze-du-jour does “S”
- What’s the relation with SBAC, HBAC, IBAC?



Q: What do “least privilege”, “well-compartimentalized” mean?

Status of Breeze

- Interpreter with lots of different flags & externals
  - standard library in different variants
  - programming experiments; exploring design space
  - compiler to SAFE architecture planned for the future
- Various subsets formalized in Coq & Isabelle
  - proved termination-insensitive non-interference
  - big-step semantics (concurrency not formalized yet)
- Ongoing effort on releasing a 1st stable version