Breeze: A Language For Writing Secure Software

Cătălin Hrițcu
Clean-slate design of Resilient, Adaptive, Secure Hosts
The SAFE team

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

Sony Makes it Official: PlayStation Network Hacked

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

The attack on DigiNotar, an issuer of SSL certificates, appears more serious than initially thought. A report by the company, which was hacked last week, indicates that many of its customers were affected.

The breach raised questions about the company’s ability to protect its infrastructure and the integrity of its certificates.

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

Smaller drones pose a threat to U.S. military operations, raising concerns about potential cyber-attacks.

On Tuesday Fairfax Media reported that Android smartphones are the target of a worm that is granted access to the phone without user permission. Once granted access, the worm can access data that was shown to be vulnerable to the worm.

The worm, known as the “Root worm,” can also access data on other smartphones connected to the same network. The worm has been found in smartphones from various manufacturers, including Samsung and HTC.

Android smartphones that are vulnerable to the worm can be infected by a malicious app that is granted access to the phone without user permission. Once granted access, the worm can access data that was shown to be vulnerable to the worm.

The worm, known as the “Root worm,” can also access data on other smartphones connected to the same network. The worm has been found in smartphones from various manufacturers, including Samsung and HTC.
# Common Weaknesses Enumeration: Top 25

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>93.8</td>
<td>CWE-89</td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>[2]</td>
<td>83.3</td>
<td>CWE-78</td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>[3]</td>
<td>79.0</td>
<td>CWE-120</td>
<td>Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')</td>
</tr>
<tr>
<td>[4]</td>
<td>77.7</td>
<td>CWE-79</td>
<td>Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')</td>
</tr>
<tr>
<td>[5]</td>
<td>76.9</td>
<td>CWE-305</td>
<td>Missing Authentication for Critical Function</td>
</tr>
<tr>
<td>[6]</td>
<td>76.8</td>
<td>CWE-862</td>
<td>Missing Authorization</td>
</tr>
<tr>
<td>[7]</td>
<td>75.0</td>
<td>CWE-798</td>
<td>Use of Hard-coded Credentials</td>
</tr>
<tr>
<td>[8]</td>
<td>75.0</td>
<td>CWE-311</td>
<td>Missing Encryption of Sensitive Data</td>
</tr>
<tr>
<td>[9]</td>
<td>74.0</td>
<td>CWE-434</td>
<td>Unrestricted Upload of File with Dangerous Type</td>
</tr>
<tr>
<td>[10]</td>
<td>73.8</td>
<td>CWE-807</td>
<td>Reliance on Untrusted Inputs in a Security Decision</td>
</tr>
<tr>
<td>[11]</td>
<td>73.1</td>
<td>CWE-250</td>
<td>Execution with Unnecessary Privileges</td>
</tr>
<tr>
<td>[12]</td>
<td>70.1</td>
<td>CWE-352</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>[13]</td>
<td>69.3</td>
<td>CWE-22</td>
<td>Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')</td>
</tr>
<tr>
<td>[14]</td>
<td>68.5</td>
<td>CWE-494</td>
<td>Download of Code Without Integrity Check</td>
</tr>
<tr>
<td>[16]</td>
<td>66.0</td>
<td>CWE-829</td>
<td>Inclusion of Functionality from Untrusted Control Sphere</td>
</tr>
<tr>
<td>[17]</td>
<td>65.5</td>
<td>CWE-732</td>
<td>Incorrect Permission Assignment for Critical Resource</td>
</tr>
<tr>
<td>[18]</td>
<td>64.6</td>
<td>CWE-676</td>
<td>Use of Potentially Dangerous Function</td>
</tr>
<tr>
<td>[19]</td>
<td>64.1</td>
<td>CWE-327</td>
<td>Use of a Broken or Risky Cryptographic Algorithm</td>
</tr>
<tr>
<td>[20]</td>
<td>62.4</td>
<td>CWE-131</td>
<td>Incorrect Calculation of Buffer Size</td>
</tr>
<tr>
<td>[21]</td>
<td>61.5</td>
<td>CWE-307</td>
<td>Improper Restriction of Excessive Authentication Attempts</td>
</tr>
<tr>
<td>[22]</td>
<td>61.1</td>
<td>CWE-601</td>
<td>URL Redirection to Untrusted Site ('Open Redirect')</td>
</tr>
<tr>
<td>[23]</td>
<td>61.0</td>
<td>CWE-134</td>
<td>Uncontrolled Format String</td>
</tr>
<tr>
<td>[24]</td>
<td>60.3</td>
<td>CWE-190</td>
<td>Integer Overflow or Wraparound</td>
</tr>
<tr>
<td>[25]</td>
<td>59.9</td>
<td>CWE-759</td>
<td>Use of a One-Way Hash without a Salt</td>
</tr>
</tbody>
</table>

*http://cwe.mitre.org/top25/index.html#Listing*
Clean slate design improves co-design facilitates Formal methods

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

Precise specifications
Global guarantees
Machine checked proofs
Breeze

• High-level PL for writing secure software
  – user programs & system services (+ security policies)
• functional core ($\lambda$) + concurrency ($\Pi$) + 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

- PCI2e 2 GB/s each direction
- DRAM (2GB)
- FPGA
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]

\[
\begin{align*}
\rho, pc \vdash \lambda x. e \downarrow (\rho, \lambda x. e)@pc \\
\rho, pc \vdash x \downarrow v@(l \lor pc)
\end{align*}
\]

\[
\begin{align*}
\rho, pc \vdash e_1 \downarrow (\rho', \lambda x. e)@l_1 \\
\rho, pc \vdash e_2 \downarrow v_2@l_2 \\
\rho'[x \mapsto v_2@l_2], pc \lor l_1 \vdash e \downarrow v_3@l_3 \\
\rho, pc \vdash e_1 e_2 \downarrow v_3@l_3
\end{align*}
\]

• Termination-insensitive

\[
\begin{align*}
\rho_1, pc \vdash e \downarrow v_1@l_1 \\
\rho_2, pc \vdash e \downarrow v_2@l_2
\end{align*}
\]

\[
\Rightarrow v_1@l_1 \simeq_l v_2@l_2
\]

non-interference

\[
\rho_1 \simeq_l \rho_2
\]
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)

- 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)

[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
  - \((P,_,_,_) = \text{newPrin} \ “P“; \ pill = 42@(P \rightarrow [P])\)
- High data can be hidden under low labels
  - \(x = [1,2,\text{pill}]@L; \text{send} \ cpub \ x\)
- IFC violations are dynamic errors
  - \(\text{recv} \ cpub \ x; \text{map} \ ((+) \ 1) \ x\)
- Labels themselves are an IF channel
  - \(x = (\text{if} \ \text{secret}@H \ \text{then} \ ()@H \ \text{else} \ ()@\text{top}); \\ \text{copy} = (\text{labelOf} \ x == H)\)
Non solution(s)

• Labels are an IF channel, so hide labels
  \[
  \rho, pc \vdash e \downarrow v@l
  \]
  \[
  \rho, pc \vdash \text{labelOf } e \downarrow l@l
  \]

• 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**

\[
\begin{align*}
\rho \vdash e, \ pc \downarrow \ v@l, \ pc' \quad l \sqsubseteq lb \quad pc' \sqsubseteq lb \\
\rho \vdash lb \langle e \rangle, \ pc \downarrow \ v@lb, \ pc
\end{align*}
\]

• 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

• 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? \( (S = \text{lexical propagation}) \)
• Q: Is authority passed from caller to callee? \( (D = \text{dynamic propagation}) \)
• 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-compartmentalized” 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