Beyond Virtual Machines: Dynamic Control-Flow Integrity

Antonio Barresi[1], Mathias Payer[2], Thomas Gross[1]

[1]ETH Zurich

[2]Purdue University

ETH Zürich

LST Laboratory for Software Technology

Memory errors

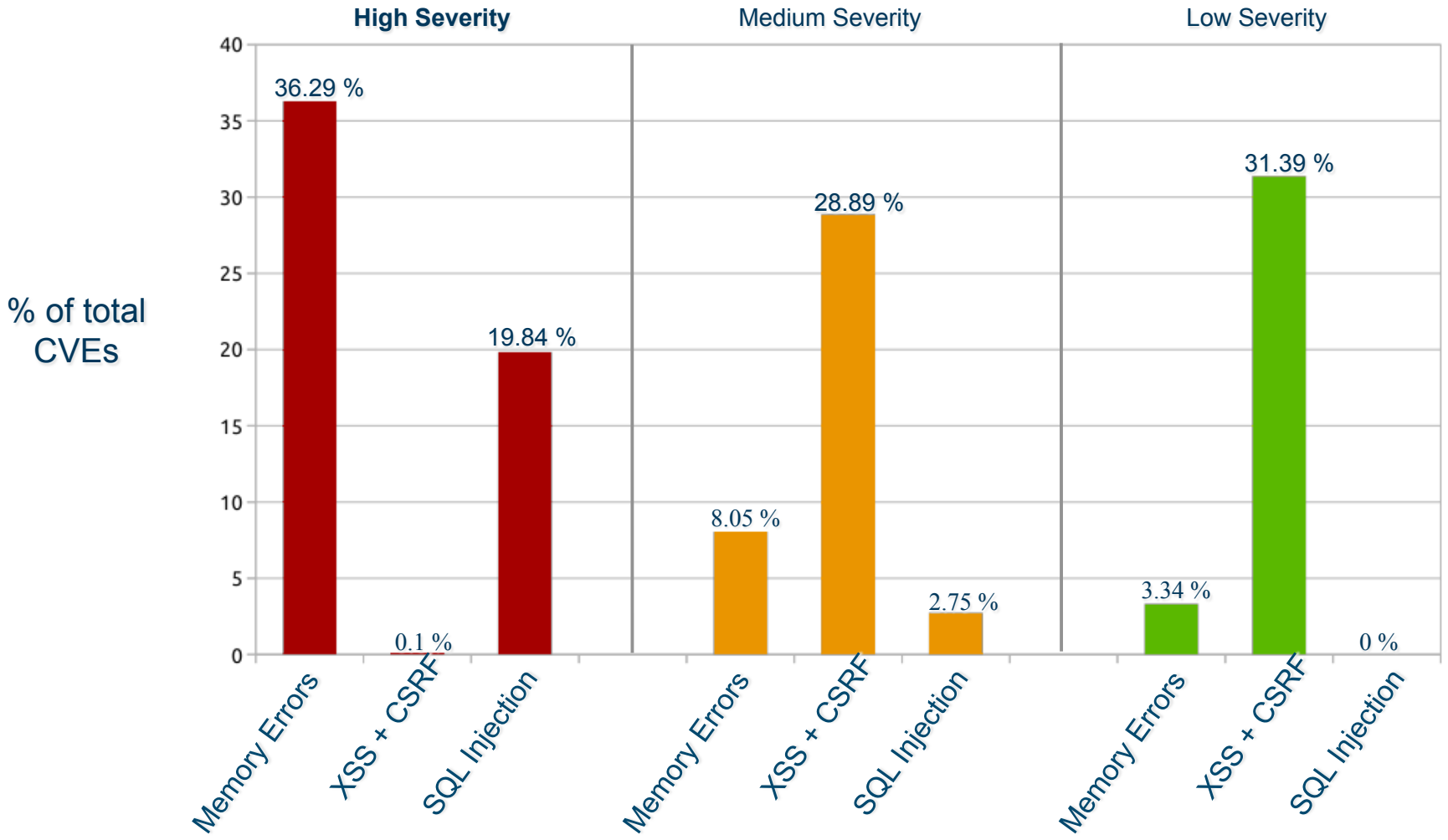
- **“Unsafe“ languages like C/C++ are still very popular**
  - Prediction: C/C++ will be with us for a long time
  - Yes, there are alternatives ..... sometimes

- **Unsafe: modification of arbitrary memory location**
- **Memory error: any corruption**

- **Memory errors can lead to serious security vulnerabilities**
  - Worst case: attackers gain arbitrary code execution capabilities

http://www.langpop.com/
http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

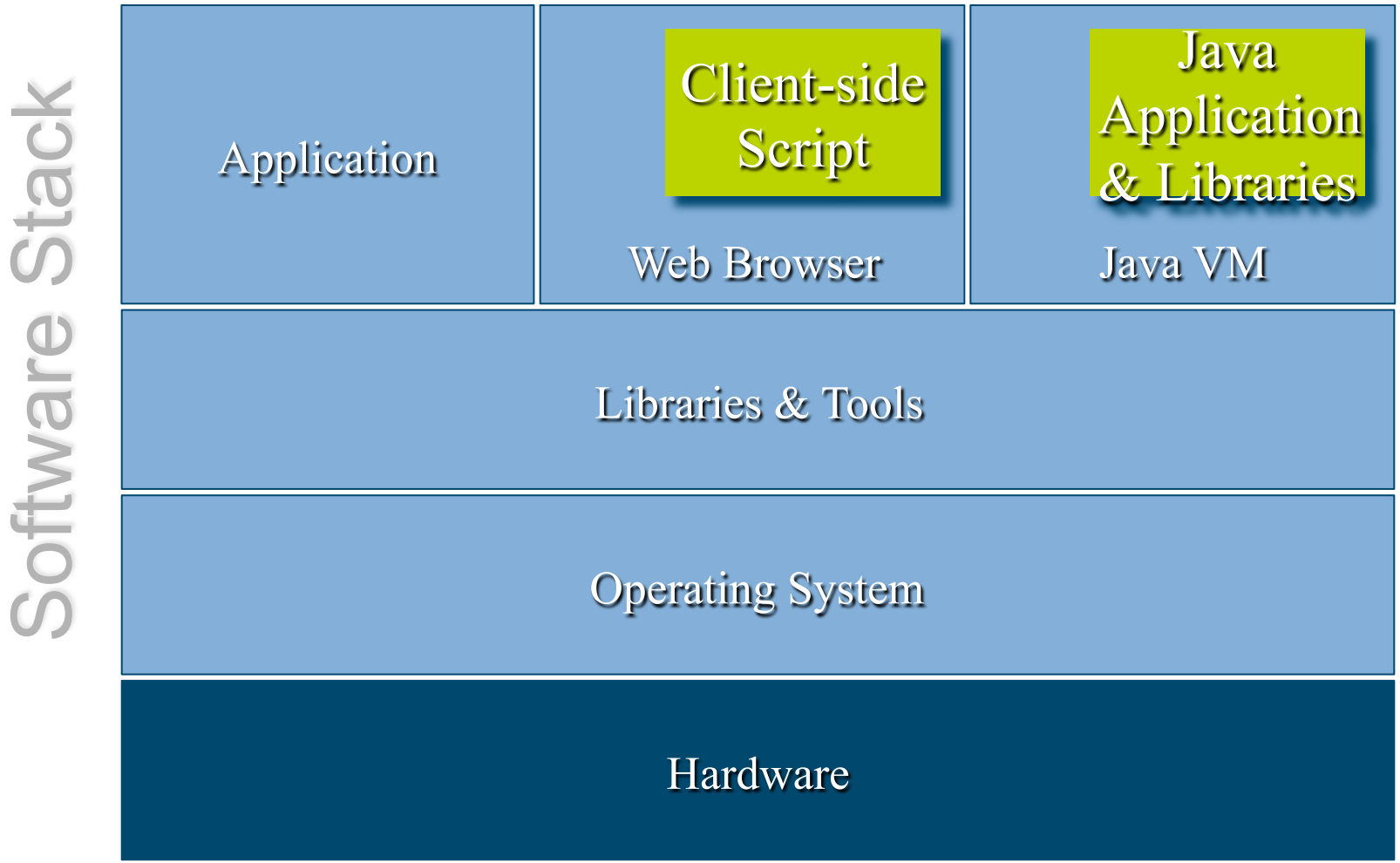
Common vulnerabilities and exposures (CVE)



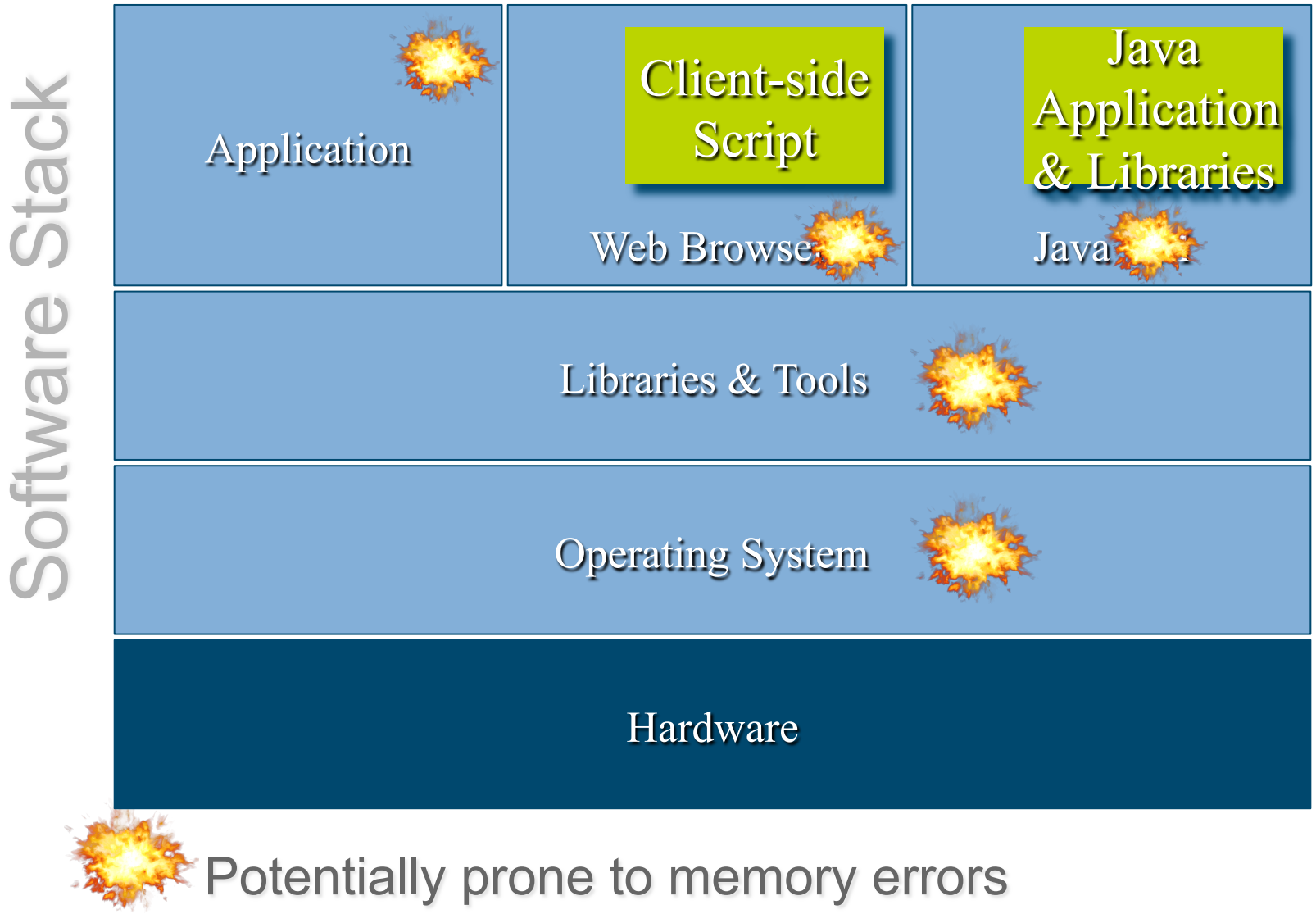
https://cve.mitre.org/

Total CVEs, with „Insufficient Information“ excluded: 25'298, time span: 1.1.2009 – 26.8.2014
Memory Errors: CWE-119, CWE-399 „use after free“, CWE-189 in High / XSS + CSRF: CWE-79, CWE-352 / SQL Injection: CWE-89

Modern software stack



Modern software stack



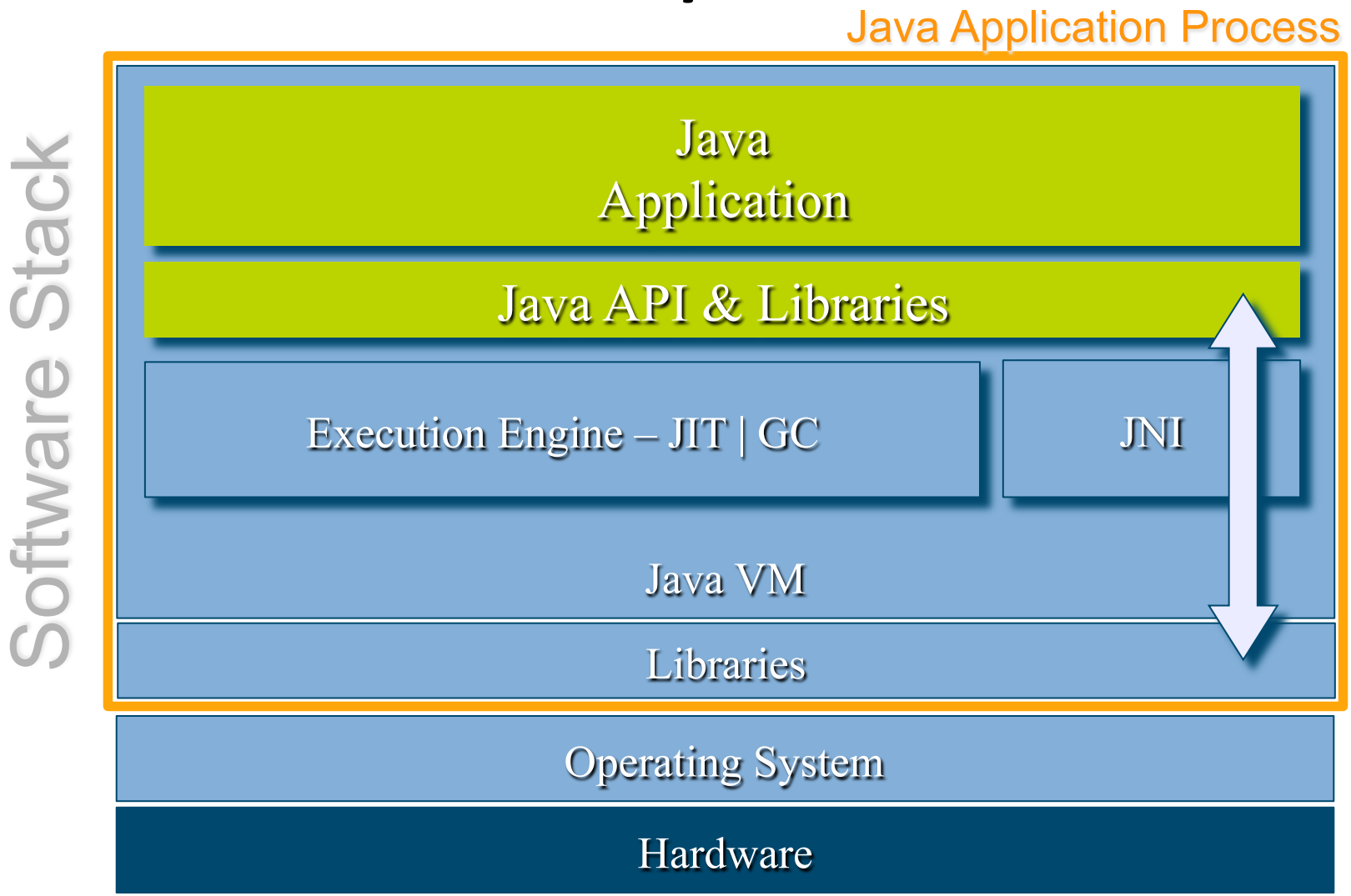
Safe languages (VM based)

- **Just use a type-safe & memory-safe language ?**
  - But language VM
    - May be implemented in an unsafe language
    - May use or provide interface to unsafe libraries
  - Memory errors are still an issue

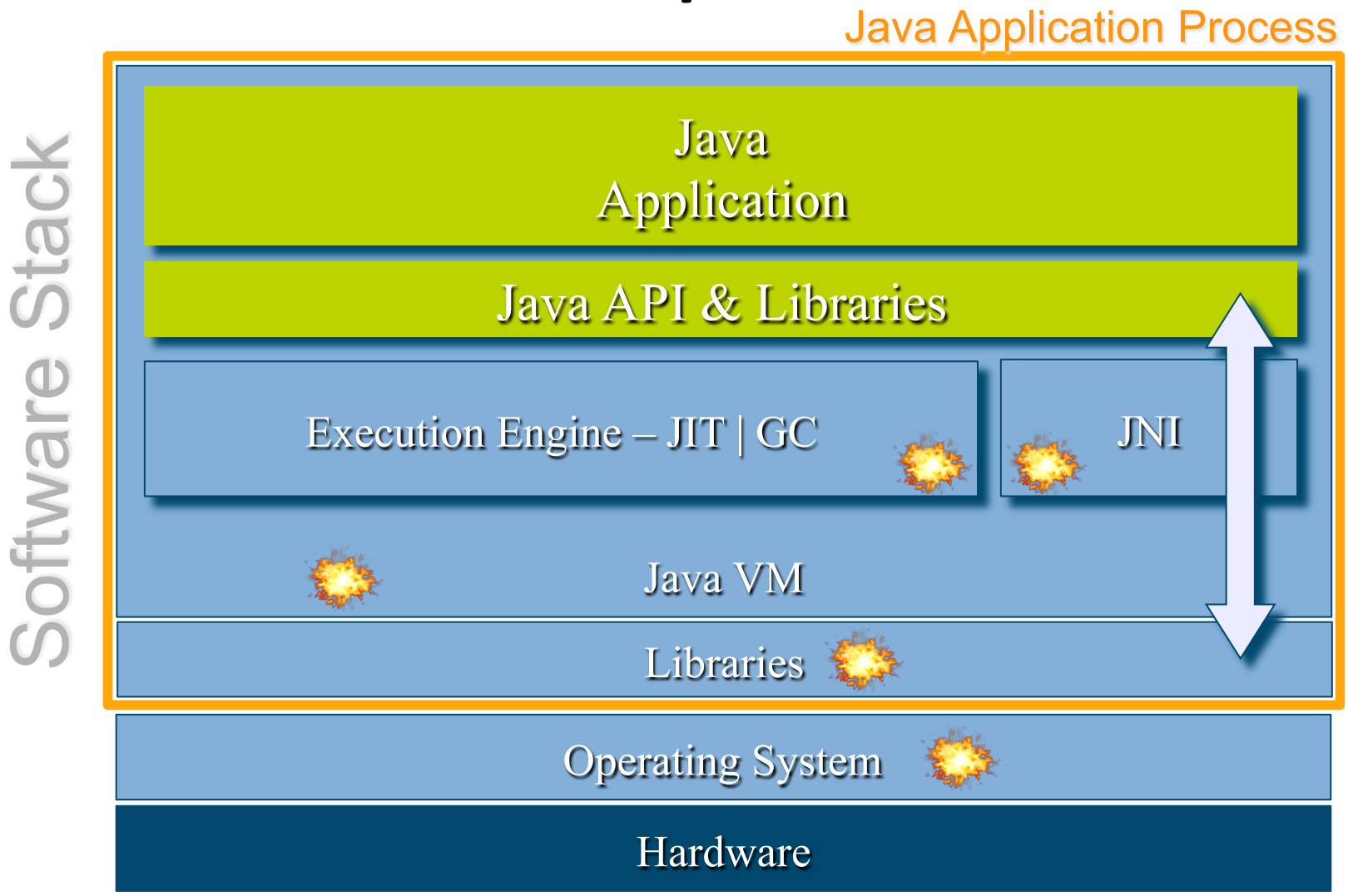
Safe languages (VM based)

- **Just use a type-safe & memory-safe language ?**
  - But language VM
    - May be implemented in an unsafe language
    - May use or provide interface to unsafe libraries
  - Memory errors are still an issue
- **Attacker may exploit memory errors**
  - In the VM
  - In unsafe libraries used by VM or application

Java VM written in C/C++



Java VM written in C/C++



Potentially prone to memory errors

Attacking safe language VMs

- **Example: Java VM**
  - CVE-2013-1491
  - Targetted Oracle Java SE 7 / 6 / 5
  - Memory error in OpenType fonts handling within native layer of JRE
    - Leveraged to arbitrary code excecution
    - Completely bypassed DEP & ASLR

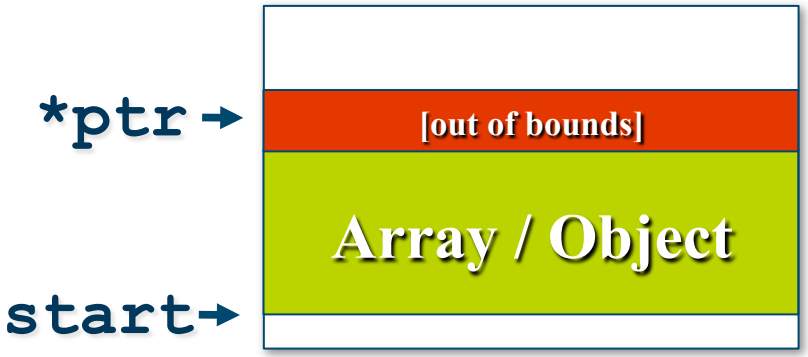
Demonstrated at Pwn2Own at CanSecWest 2013 by Joshua Drake (on Windows 8 + Java SE 7 Update 17)
http://www.accuvant.com/blog/pwn2own-2013-java-7-se-memory-corruption
https://media.blackhat.com/bh-ad-11/Drake/bh-ad-11-Drake-Exploiting_Java_Memory_Corruption-WP.pdf

“Unsafe“ languages

- **Allow low-level access to memory**
  - Typed pointers & pointer arithmetic
  - No automatic bounds checking or index checking
- **Weakly enforced typing**
  - Cast (almost) anything to pointers
- **Explicit memory management**
  - Like malloc() & free() in C

Types of memory errors

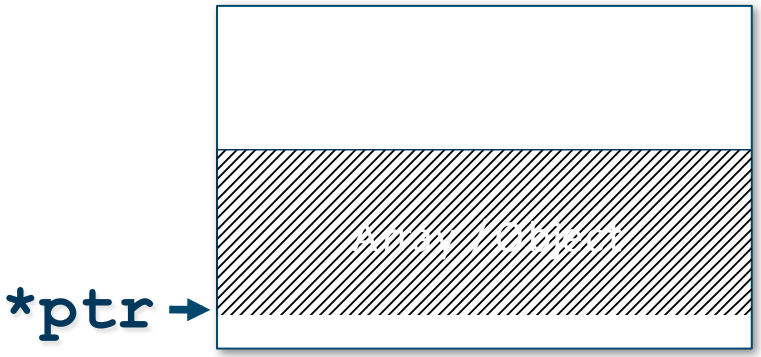
- **Spatial error**



De-reference pointer that is out of bounds

Read or Write operation

- **Temporal error**

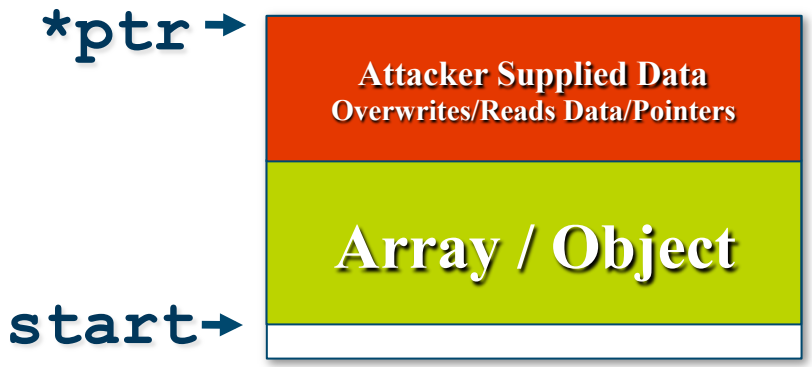


De-reference pointer to freed memory

Read operation

Exploiting memory errors

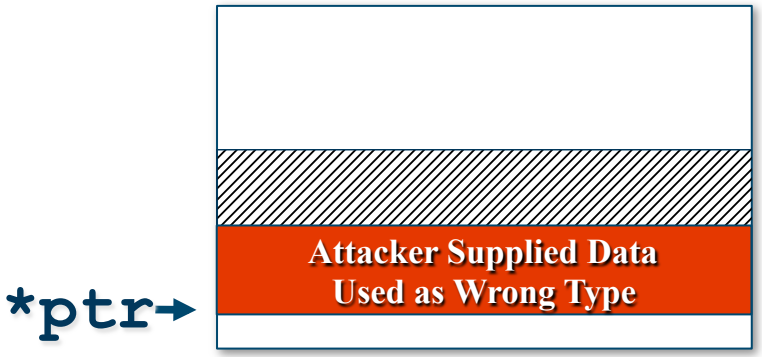
- **Spatial error**



Overwrite data or pointers

Used or de-referenced later

- **Temporal error**



Make application allocate memory in the freed area

Used as old type

Attackers use memory errors to

- **Overwrite data or pointers**
  - Function pointers, sensitive data, index values, etc.
- **Mislead information**
  - E.g., corrupt a length field
- **Construct attacker primitives**
  - Write primitive (write any value to arbitrary address)
  - Read primitive (read from any address)

Attack types

- **Code corruption attack**
- **Control-flow hijack attack**
- **Data-only attack**
- **Information leak**

Attack model according to: „sok: eternal war in memory“ laszlo szekeres, mathias payer, tao wei, dawn song
Http://www.Cs.Berkeley.Edu/~dawnsong/papers/oakland13-sok-cr.Pdf

Attack types

- Code corruption attack
- **Control-flow hijack attack**
- Data-only attack
- Information leak

Attack model according to: „sok: eternal war in memory“ laszlo szekeres, mathias payer, tao wei, dawn song
Http://www.Cs.Berkeley.Edu/~dawnsong/papers/oakland13-sok-cr.Pdf

Control-flow hijack attacks

- **Most powerful attack**

- **Hijack control-flow**
  - To attacker-supplied arbitrary machine code
  - To existing code (code-reuse attack)

- **Corrupt code pointers**
  - Return addresses, function pointers, vtable entries, exception handlers, jmp_bufs

Concluding remarks

- **Safe-language VMs a big step towards trustworthy computing platform**
- **Safe-language VMs not an island – unsafe languages remain important**
  - Attackers find exploits
  - "Benefit": attack when application assumes safety
- **Control-flow integrity protects program execution paths**
  - Static CFI elegant but not practical
- **Dynamic CFI offers chance to block wide avenue**
  - More work needed
  - Implementation
  - Evaluation models

Thank you for your attention