CS 576 – Systems Security
Data Execution Prevention

Georgios (George) Portokalidis

Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable

Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable

- Can be eliminated if we introduce the following policy

W^X Policy

The Write XOR Execute (W^X) policy mandates that in a program there are no memory pages that are both writable and executable

Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable

- Can be eliminated if we introduce the following policy

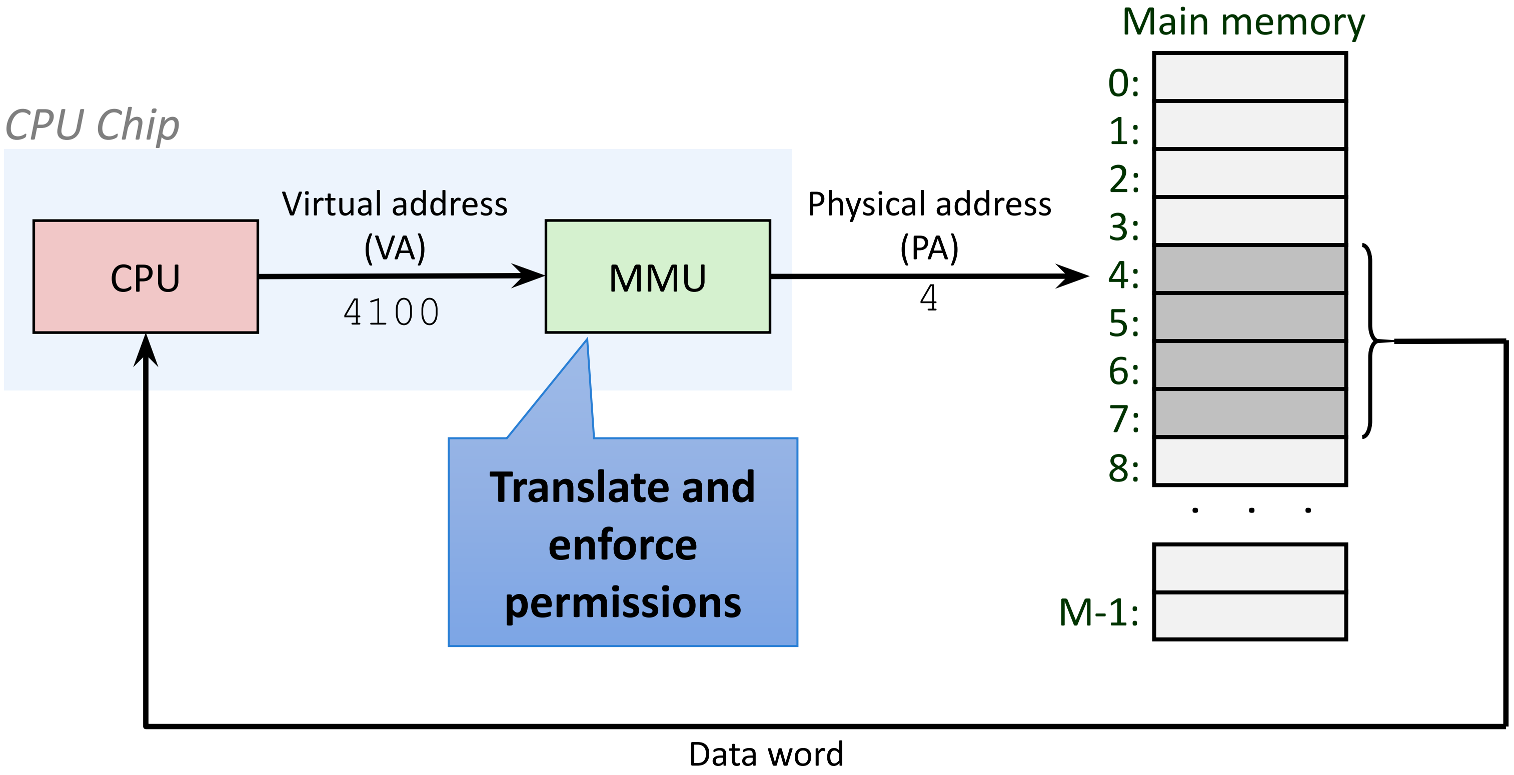
W^X Policy

How?

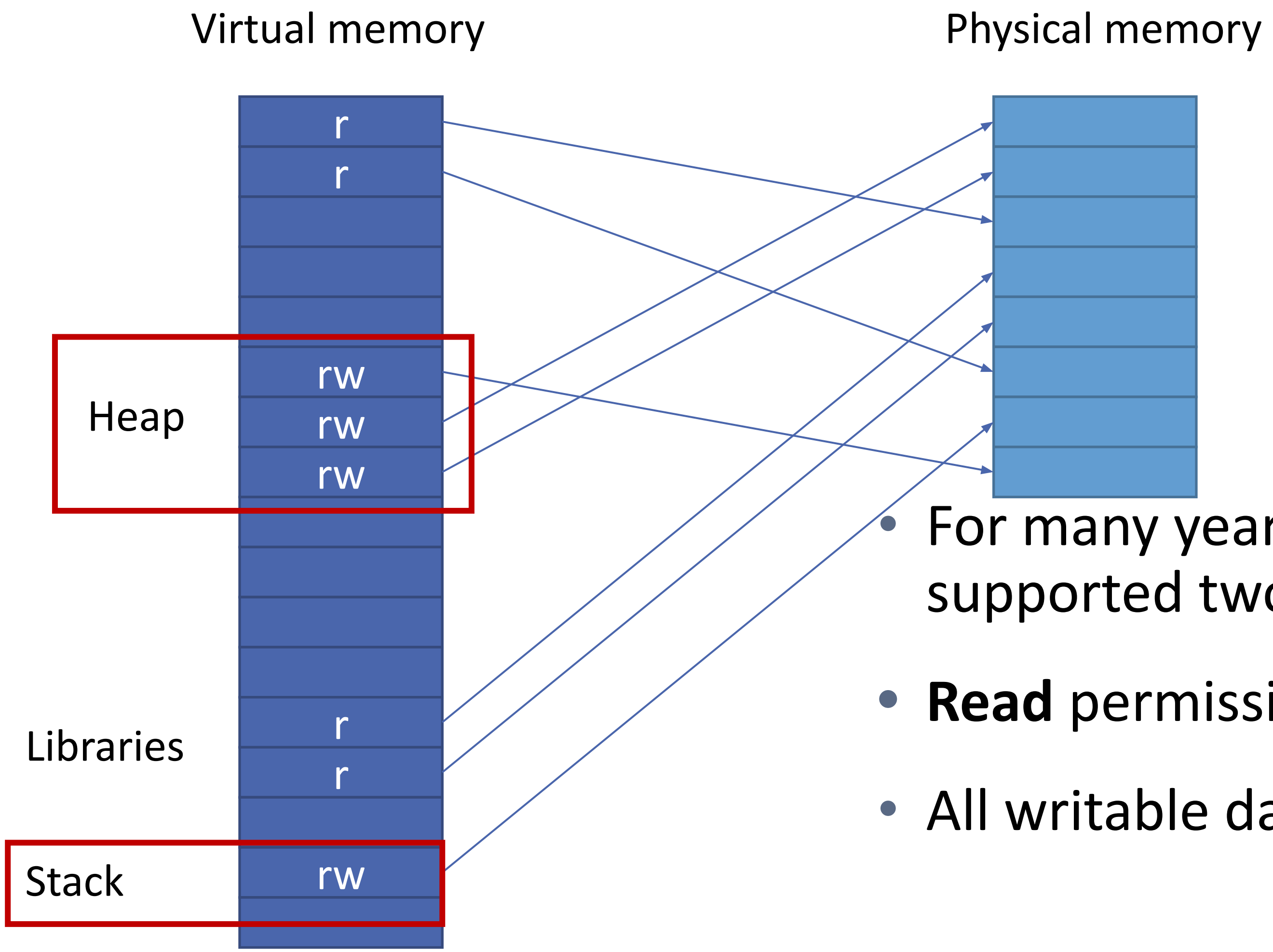
The Write XOR Execute (W^X) policy mandates that in a program there are no memory pages that are both writable and executable

The Memory Management Unit (MMU) - Paging

- Used in all modern servers, laptops, and smart phones
- One of the great ideas in computer science



Page Permissions

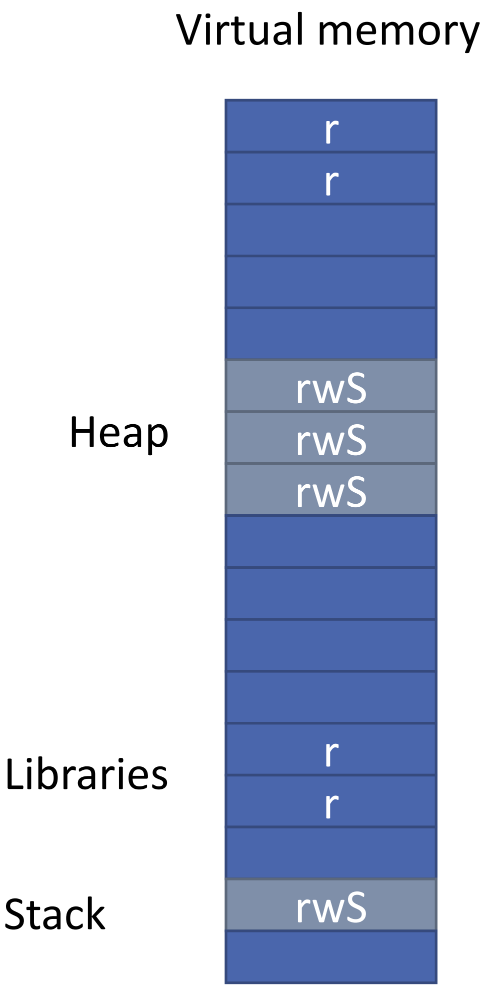


- For many years (<2000), processors supported two permissions (bits)
- **Read** permission implied **Execute**
- All writable data segments violated W^X

Early Approaches: PAGEEXEC

- A Linux kernel patch emulating non-executable memory
- Introduced in 2000 by the PaX team
- PAGEEXEC refused code execution on selected writable pages
  - Heap and stack

Emulating Non-Executable Memory



- Mark writable pages so that access causes a page fault
  - Not present □ a page-fault will be raised on every access
  - With supervisor bit (S) □ Access only allowed from the kernel
- Custom page-fault handler intercepts and checks accesses:
  - Fault caused by other instruction □ data access □ OK
  - Faulting address is being executed □ code execution □ Violation

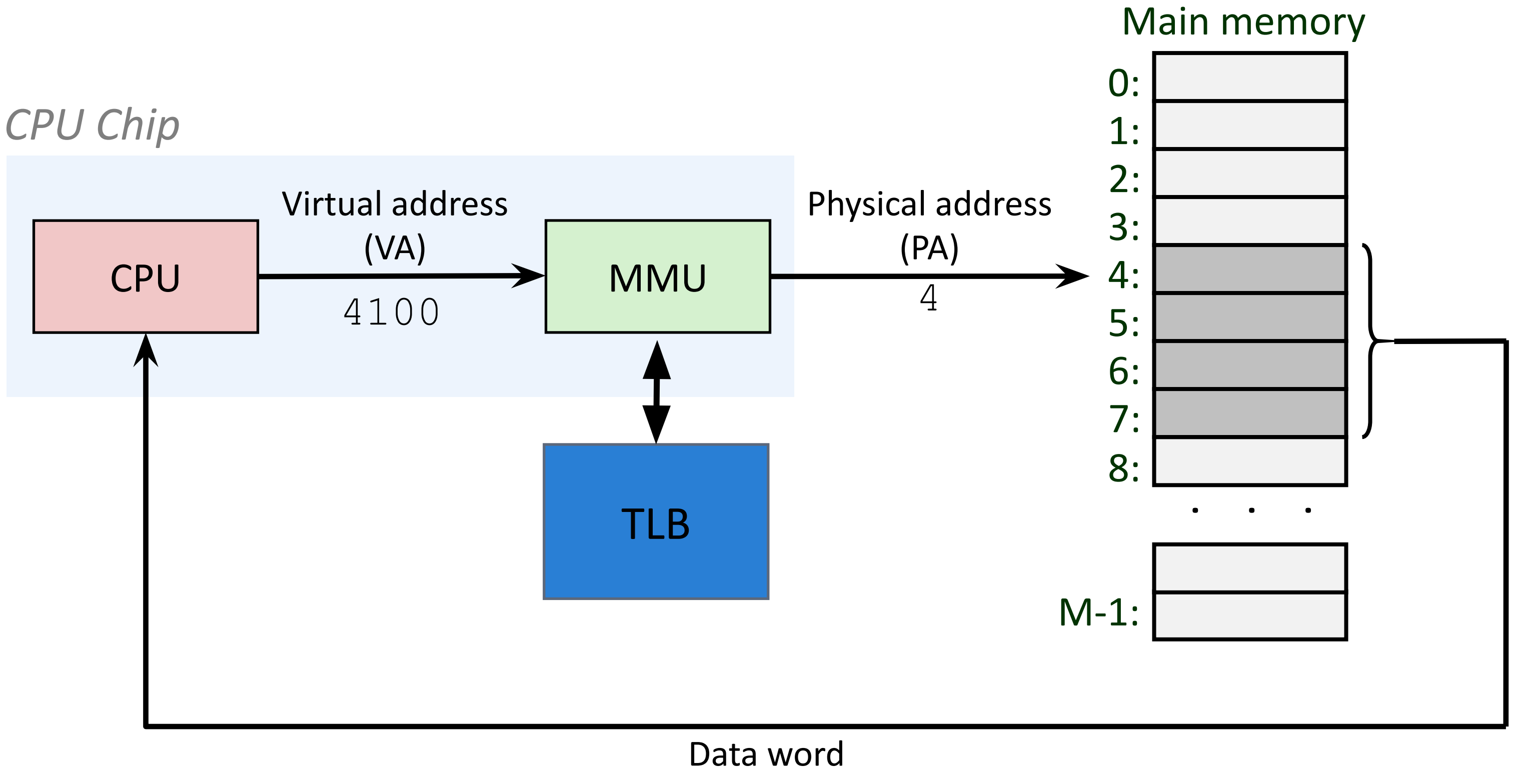
Emulating Non-Executable Memory

Should be very expensive….

- Mark writable pages so that access causes a page fault
  - Not present □ a page-fault will be raised on every access
  - With supervisor bit (S) □ Access only allowed from the kernel
- Custom page-fault handler intercepts and checks accesses:
  - Fault caused by other instruction □ data access □ OK
  - Faulting address is being executed □ code execution □ Violation

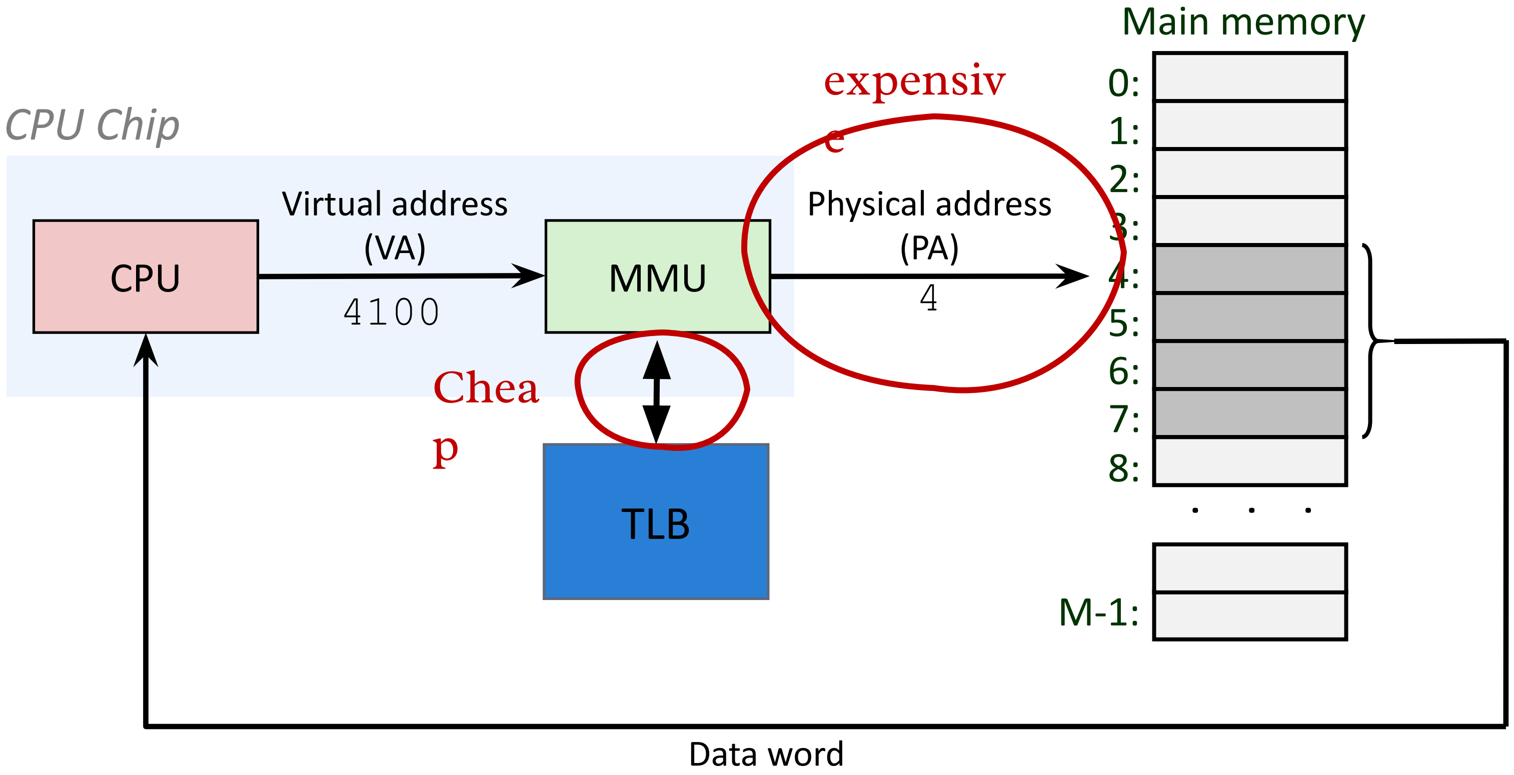
Translation Lookaside Buffer (TLB)

- A cache for storing the translations for the most frequently accessed pages



Translation Lookaside Buffer (TLB)

- A cache for storing the translations for the most frequently accessed pages



Split TLBs

- Instruction TLB (ITLB) used when fetching bytes to be decoded and executed as an instruction
  - PC □ memory `addr`
  - `addr` □ ITLB
- Data TLB (DTLB) used when reading/write bytes required by the executing instruction
  - `(addr)` -> memory `addr`
    - Example: mov (addr), reg
  - `addr` -> DTLB

Split TLBs and PAGEEXEC

- Fault caused because PC points within data area □ Violation
- Fault caused by other access
  - Remove supervisor bit from page
  - Complete load which will be added to the DTLB
  - Add supervisor bit to page
  - Subsequent accesses to address will be served by the DTLB
    - Until it is flushed or the entry for the address evicted

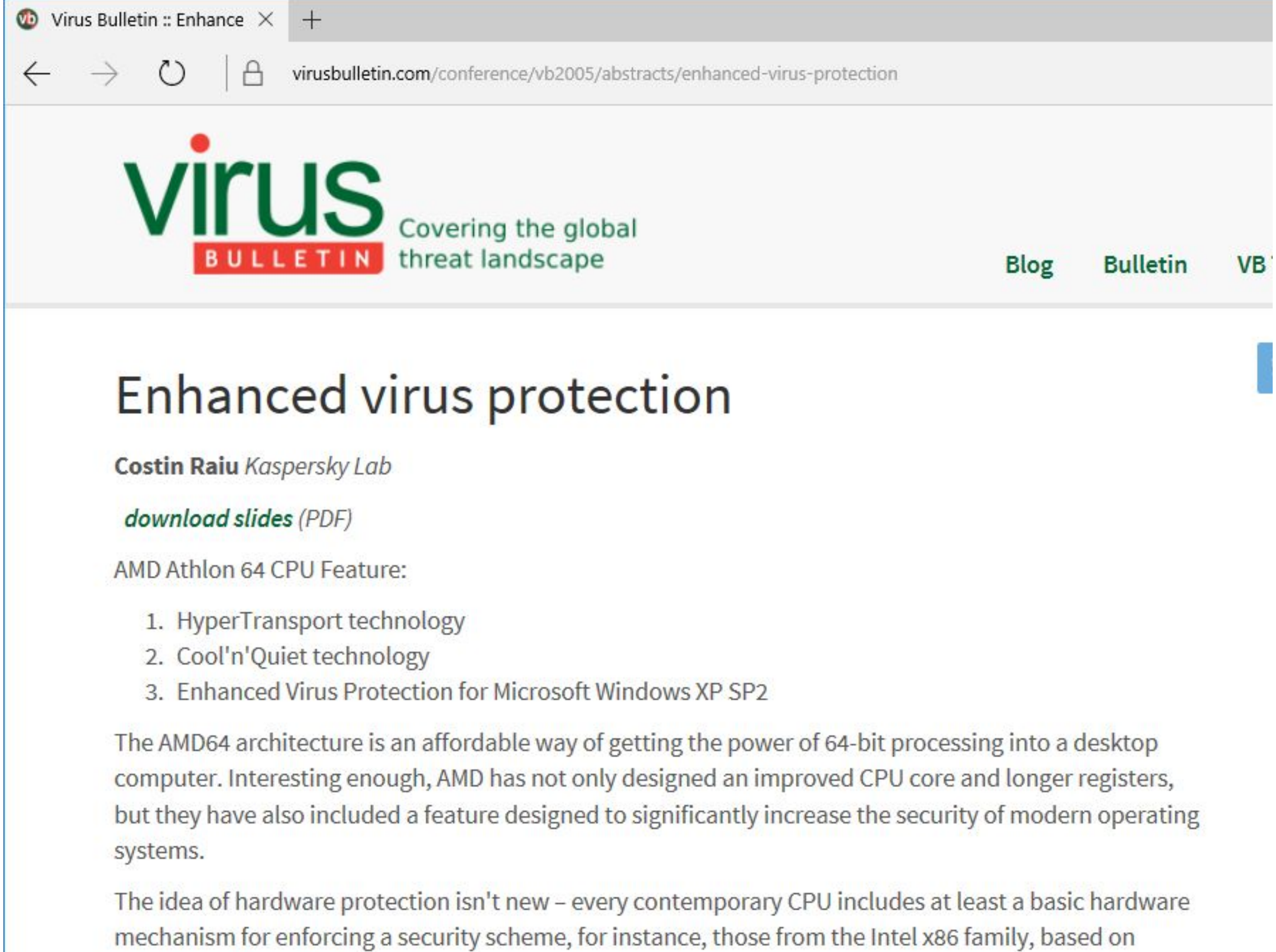
Hardware Support: NX-bit

- Processor manufacturers introduced a new bit in page permissions to prevents code injections

- Coined **N**o-e**X**ecute or **E**xecute **N**ever

- The NX-bit (No-eXecute) was introduced first by AMD to resolve such issues in 2001
  - Asserting NX, makes a readable page non-executable
  - Frequently referred to as Data Execution Prevention (DEP) on Windows

- **Marketed as antivirus technology**


Virus Bulletin :: Enhance
virusbulletin.com/conference/vb2005/abstracts/enhanced-virus-protection
virus BULLETIN
Covering the global threat landscape
Blog
Bulletin
VB

Enhanced virus protection

Costin Raiu *Kaspersky Lab*

download slides *(PDF)*

AMD Athlon 64 CPU Feature:

1. HyperTransport technology
2. Cool'n'Quiet technology
3. Enhanced Virus Protection for Microsoft Windows XP SP2

The AMD64 architecture is an affordable way of getting the power of 64-bit processing into a desktop computer. Interesting enough, AMD has not only designed an improved CPU core and longer registers, but they have also included a feature designed to significantly increase the security of modern operating systems.

The idea of hardware protection isn't new – every contemporary CPU includes at least a basic hardware mechanism for enforcing a security scheme, for instance, those from the Intel x86 family, based on

Adoption

- A non-executable stack was not immediately adopted
- The OS occasionally needed to place code in the stack
    - For example, trampoline code for handling UNIX signals
- Widely adopted today

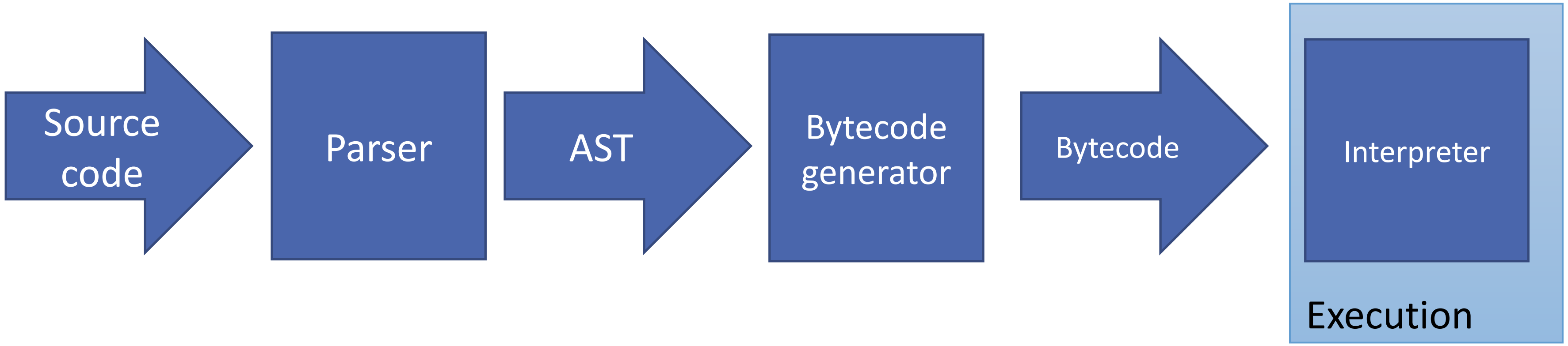
Unless You Are a Browser…

- Very popular software
  - Probably installed on every client device

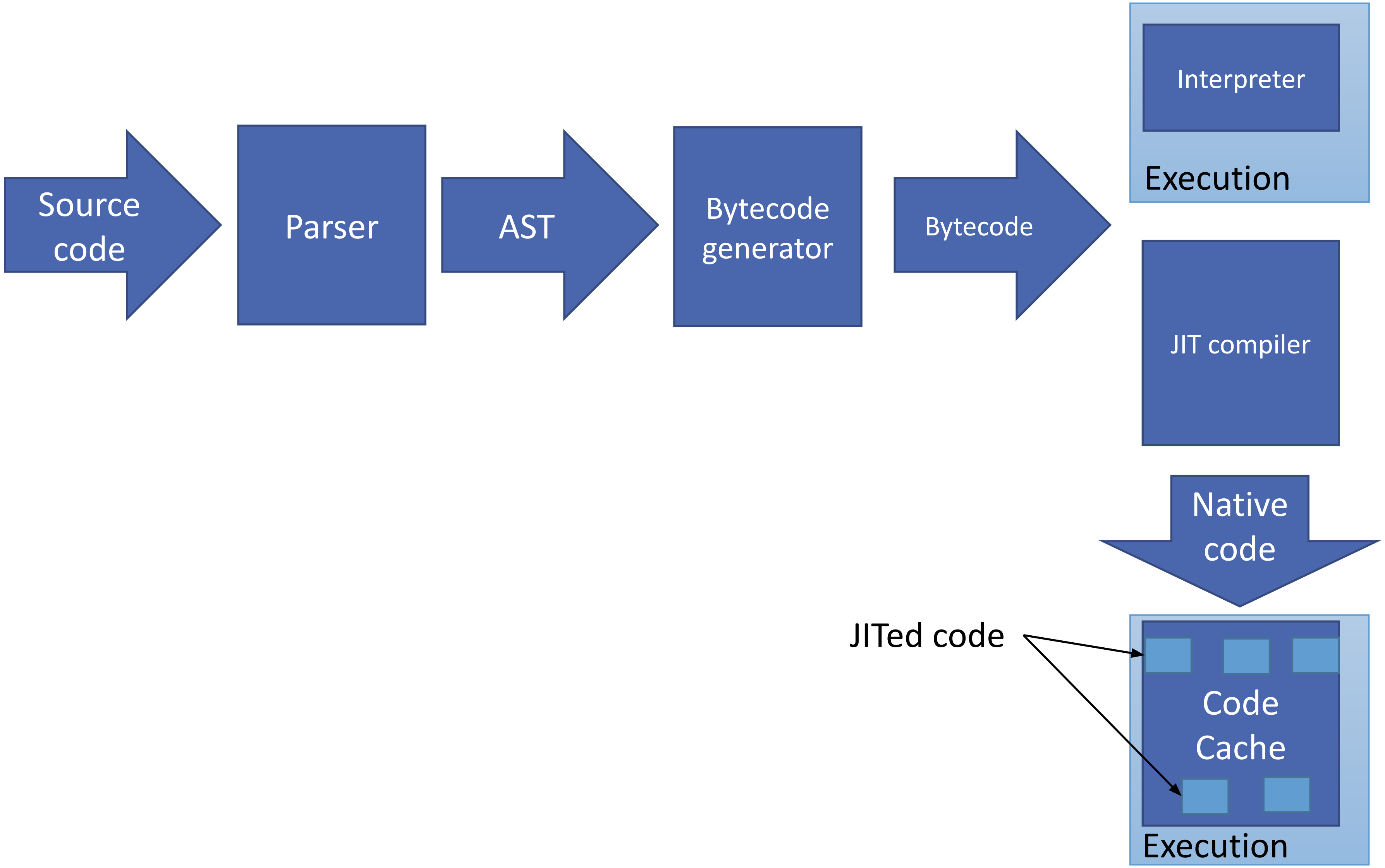
- Large and complex software

- Execute JavaScript

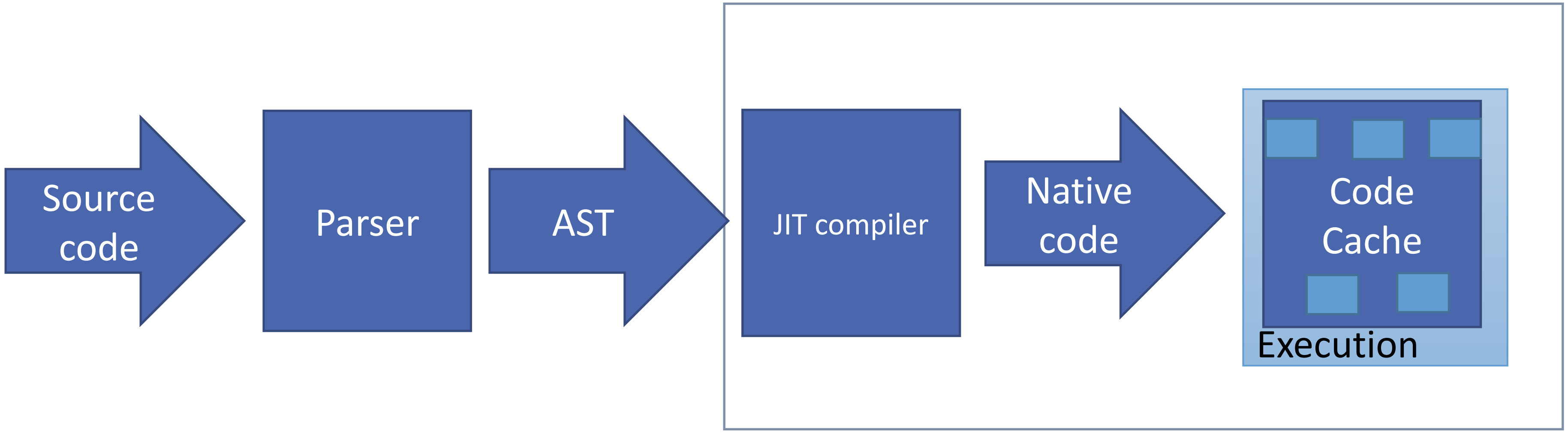
How Does JavaScript Run



How Does JavaScript Run



How Does JavaScript Run

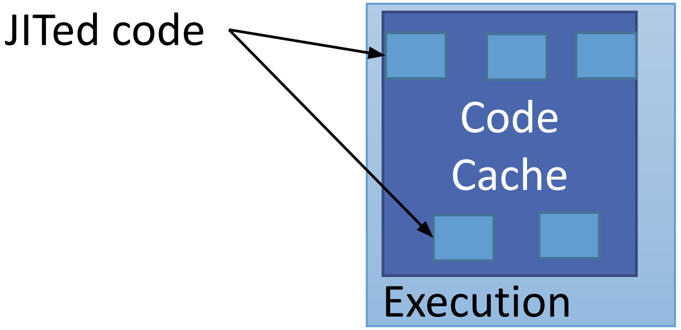


- Google V8 designed specifically to execute at speed.
- Bytecode generation skipped
- Directly emit native code
- Overall JavaScript execution improved by 150%

Code Cache

- JITed code and code cache have interesting properties from the perspective of the attacker
    - Code is continuously generated
    - Code needs to be executable

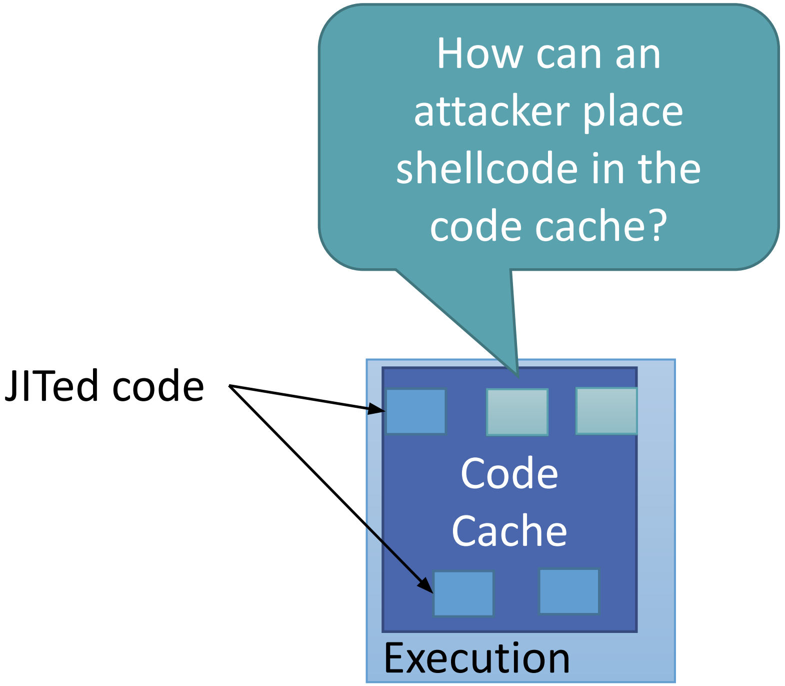
- **Violates the W^X policy**



Code Cache

- JITed code and code cache have interesting properties from the perspective of the attacker
  - Code is continuously generated
  - Code needs to be executable

- **Violates the W^X policy**



From JS to Code Cache

- JS code is JITed and placed in the code cache
- Some JS engines do not separate data and code

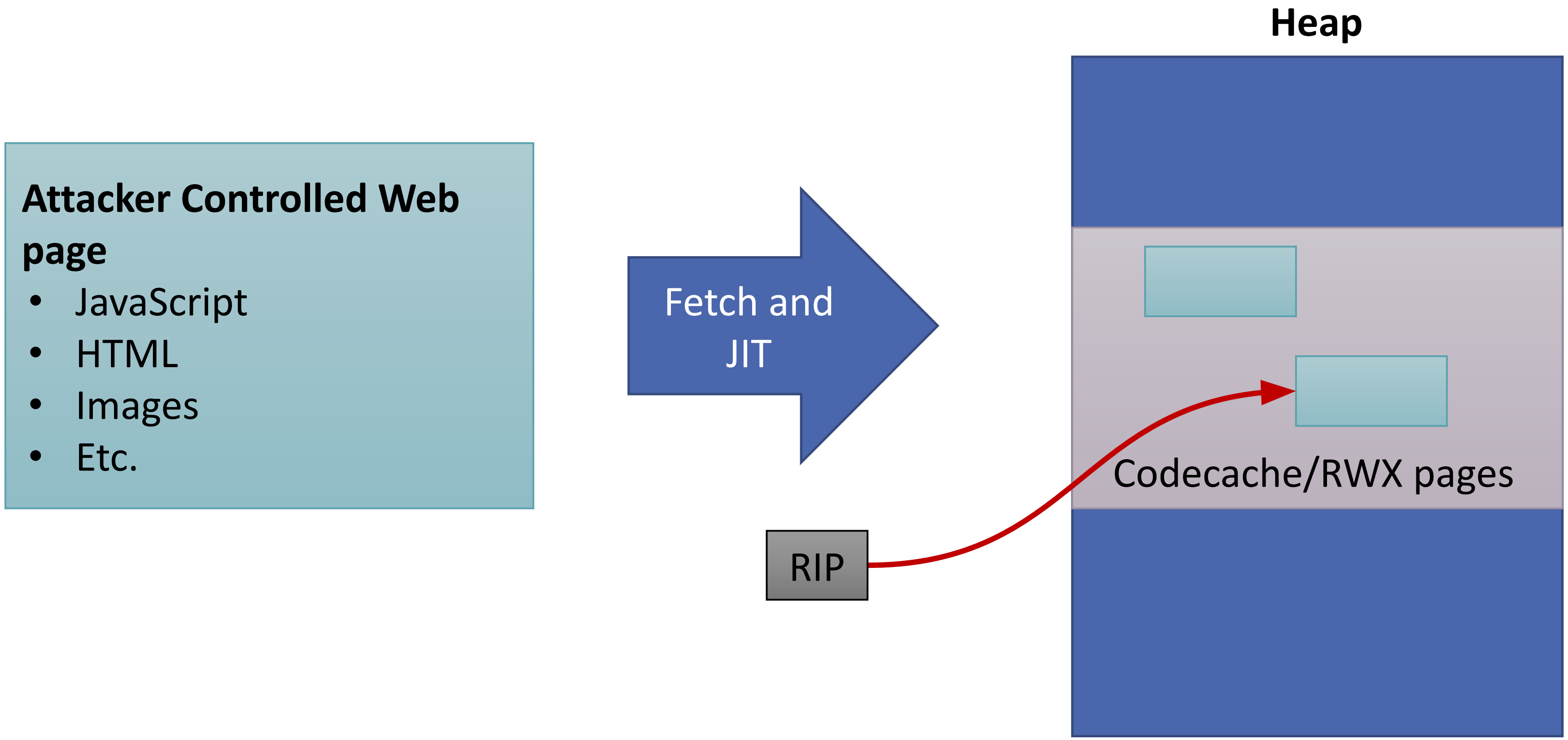
```
<html>
<body>
<script language='javascript'>

var myvar = unescape('%u\4F43%u\4552'); // CORE
myvar += unescape('%u\414C%u\214E'); // LAN!
alert("allocation done");

</script>
</body>
</html>
```

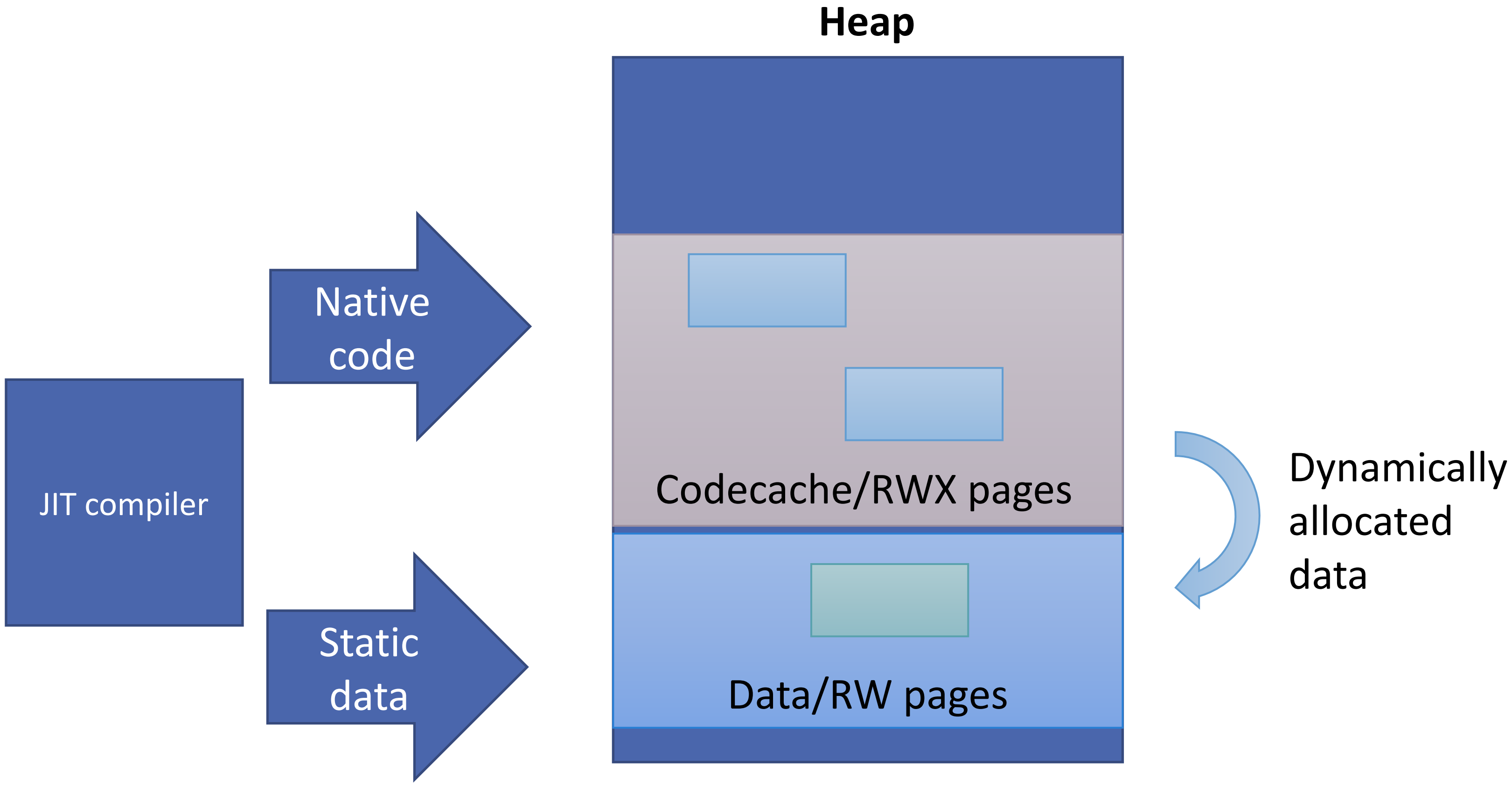
Code-Injection Attacks Against Browsers

- Return to code injected in the codecache



Avoiding Code Injection in Browsers

- Separate code and data into separate memory areas
- Still violates W^X



Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable

Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable
- Can be eliminated if we introduce the following policy

W^X Policy

The Write XOR Execute (W^X) policy mandates that in a program there are no memory pages that are both writable and executable

Writable and Executable Memory

- Code injection is possible because there is a memory area that is both writable and executable

- Can be eliminated if we introduce the following policy

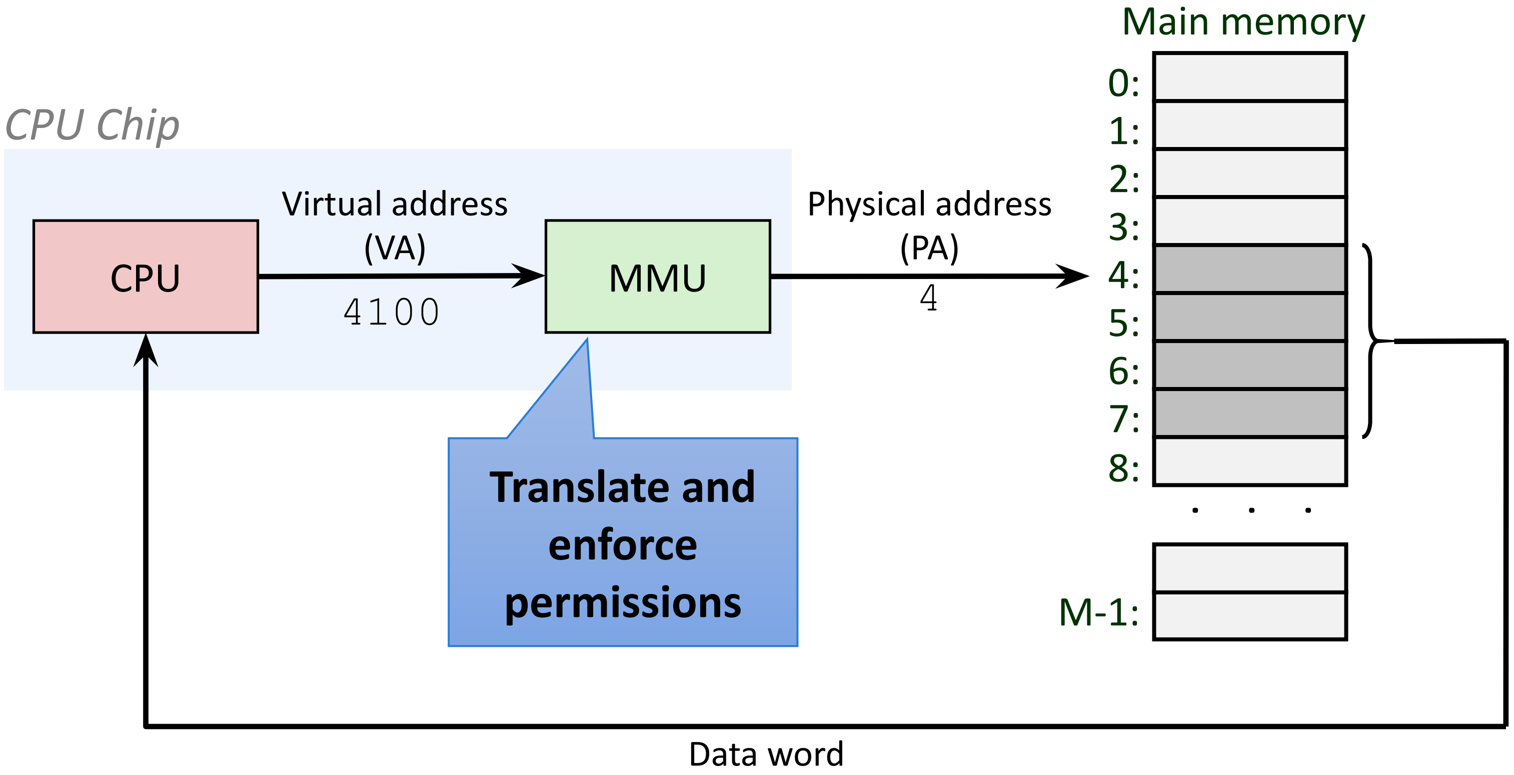
W^X Policy

How?

The Write XOR Execute (W^X) policy mandates that in a program there are no memory pages that are both writable and executable

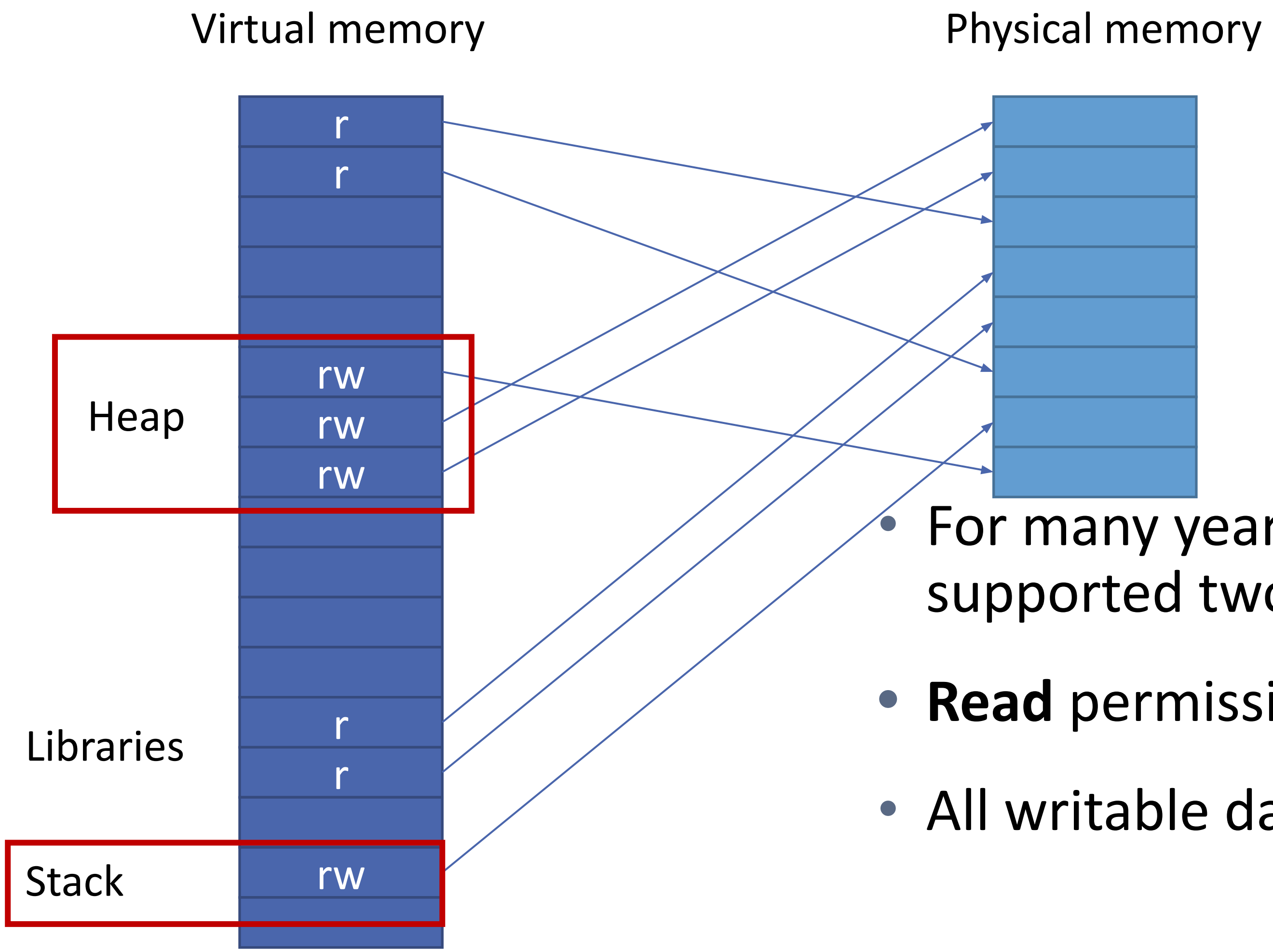
The Memory Management Unit

- Used in all modern servers, laptops, and smart phones
- One of the great ideas in computer science



Translate and enforce permissions

Page Permissions

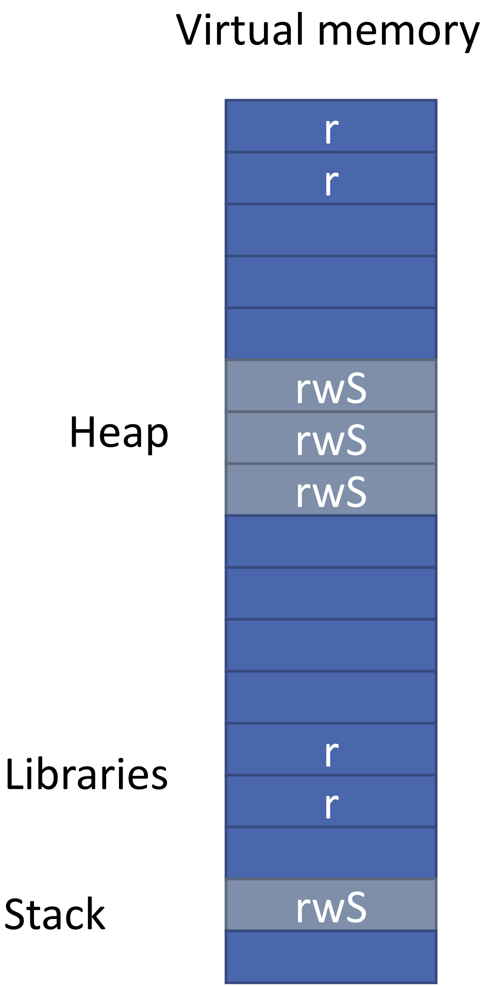


- For many years (<2000), processors supported two permissions (bits)
- **Read** permission implied **Execute**
- All writable data segments violated W^X

Early Approaches: PAGEEXEC

- A Linux kernel patch emulating non-executable memory
- Introduced in 2000 by the PaX team
- PAGEEXEC refused code execution on selected writable pages
  - Heap and stack

Emulating Non-Executable Memory



- Mark writable pages so that access causes a page fault
  - Not present □ a page-fault will be raised on every access
  - With supervisor bit (S) □ Access only allowed from the kernel
- Custom page-fault handler intercepts and checks accesses:
  - Fault caused by other instruction □ data access □ OK
  - Faulting address is being executed □ code execution □ Violation

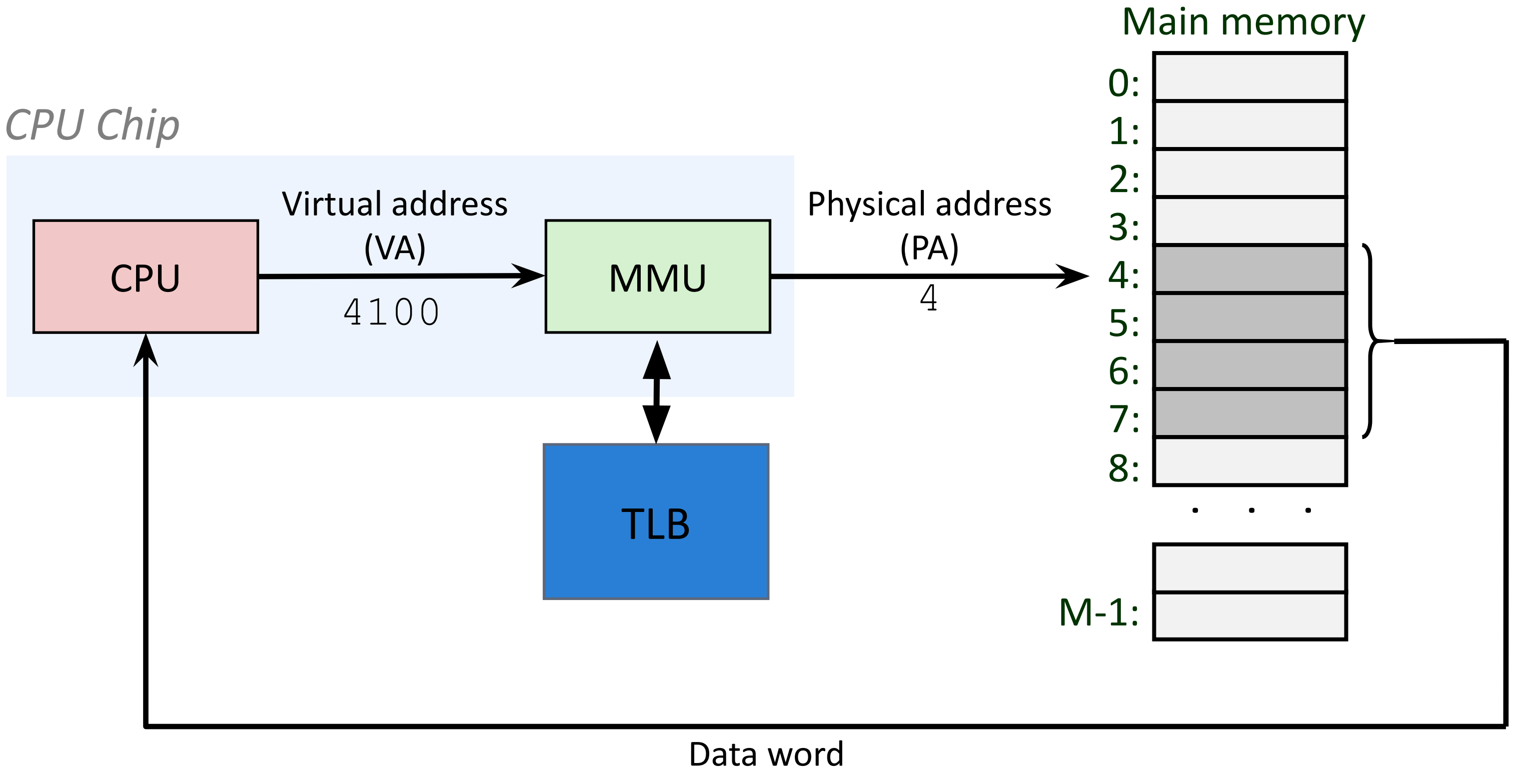
Emulating Non-Executable Memory

Should be very expensive….

- Mark writable pages so that access causes a page fault
  - Not present □ a page-fault will be raised on every access
  - With supervisor bit (S) □ Access only allowed from the kernel
- Custom page-fault handler intercepts and checks accesses:
  - Fault caused by other instruction □ data access □ OK
  - Faulting address is being executed □ code execution □ Violation

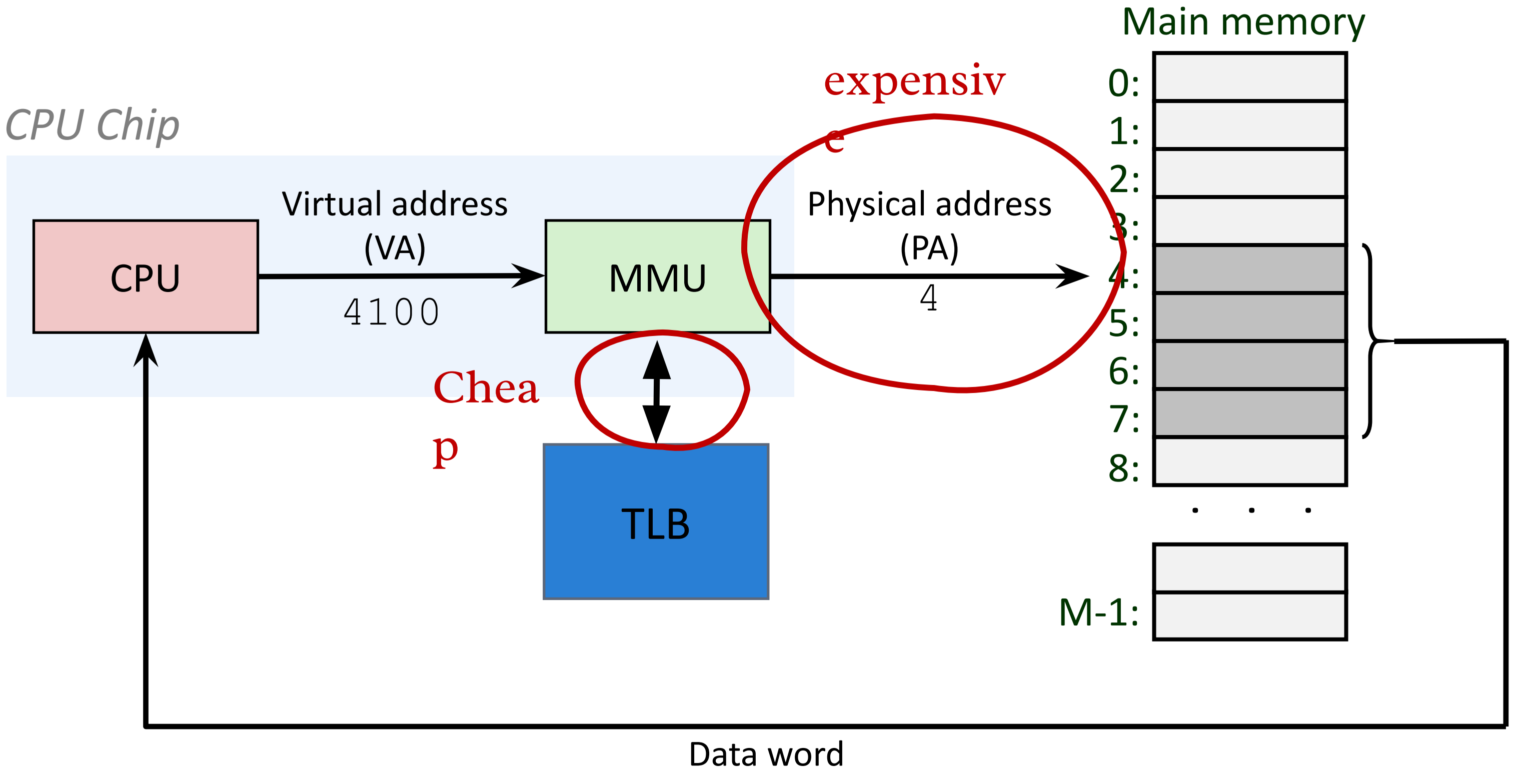
Translation Lookaside Buffer (TLB)

- A cache for storing the translations for the most frequently accessed pages



Translation Lookaside Buffer (TLB)

- A cache for storing the translations for the most frequently accessed pages



Split TLBs

- Instruction TLB (ITLB) used when fetching bytes to be decoded and executed as an instruction
  - PC □ memory `addr`
  - `addr` □ ITLB
- Data TLB (DTLB) used when reading/write bytes required by the executing instruction
  - `(addr)` -> memory `addr`
    - Example: mov (addr), reg
  - `addr` -> DTLB

Split TLBs and PAGEEXEC

- Fault caused because PC points within data area Violation

- Fault caused by other access
  - Remove supervisor bit from page
  - Complete load which will be added to the DTLB
  - Add supervisor bit to page
  - Subsequent accesses to address will be served by the DTLB
    - Until it is flushed or the entry for the address evicted

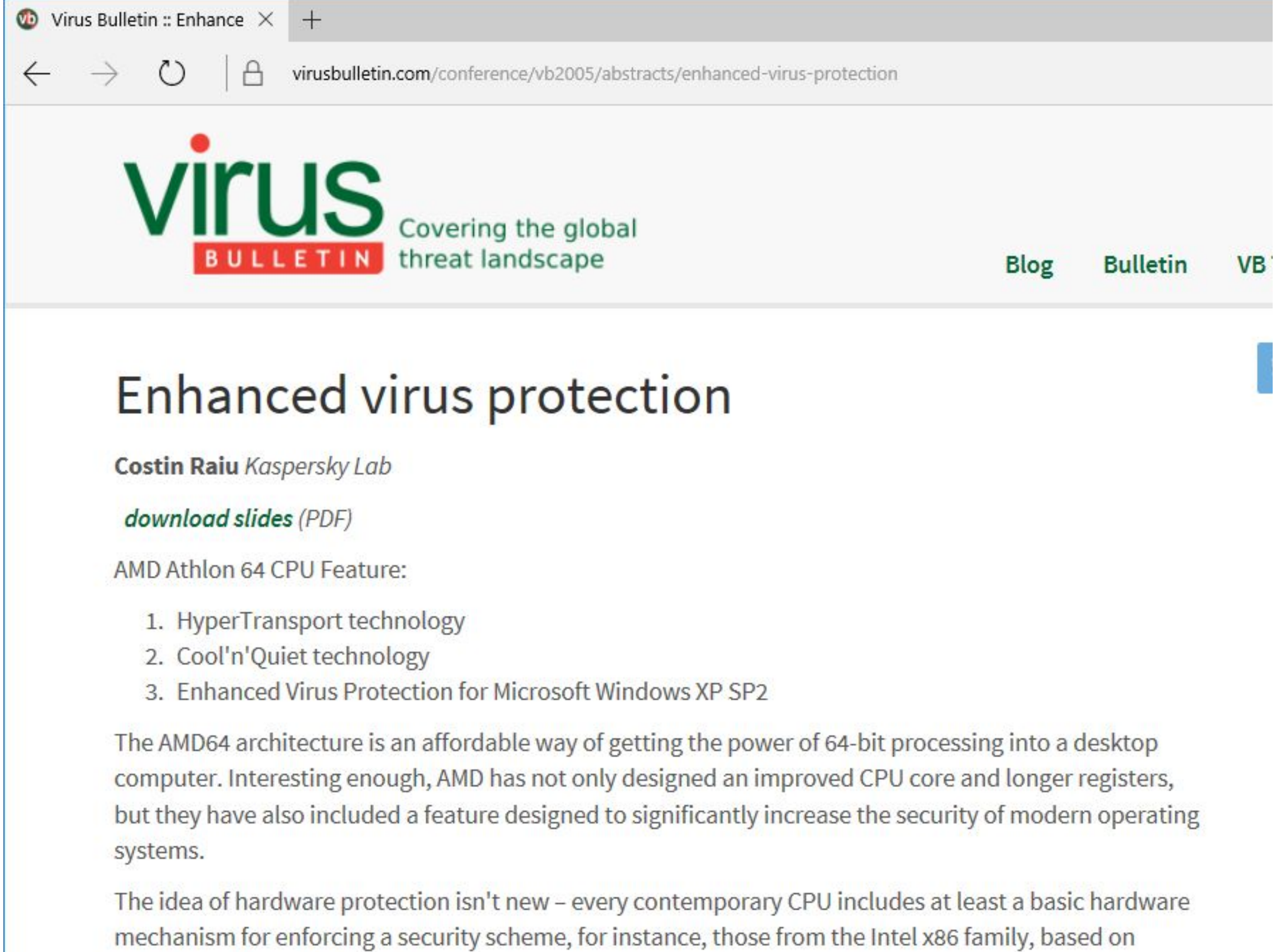
Hardware Support: NX-bit

- Processor manufacturers introduced a new bit in page permissions to prevents code injections

- Coined **N**o-e**X**ecute or **E**xecute **N**ever

- The NX-bit (No-eXecute) was introduced first by AMD to resolve such issues in 2001
  - Asserting NX, makes a readable page non-executable
  - Frequently referred to as Data Execution Prevention (DEP) on Windows

- **Marketed as antivirus technology**


Virus Bulletin :: Enhance
virusbulletin.com/conference/vb2005/abstracts/enhanced-virus-protection
virus
BULLETIN
Covering the global threat landscape
Blog
Bulletin
VB

Enhanced virus protection

Costin Raiu *Kaspersky Lab*

download slides *(PDF)*

AMD Athlon 64 CPU Feature:

1. HyperTransport technology
2. Cool'n'Quiet technology
3. Enhanced Virus Protection for Microsoft Windows XP SP2

The AMD64 architecture is an affordable way of getting the power of 64-bit processing into a desktop computer. Interesting enough, AMD has not only designed an improved CPU core and longer registers, but they have also included a feature designed to significantly increase the security of modern operating systems.

The idea of hardware protection isn't new – every contemporary CPU includes at least a basic hardware mechanism for enforcing a security scheme, for instance, those from the Intel x86 family, based on

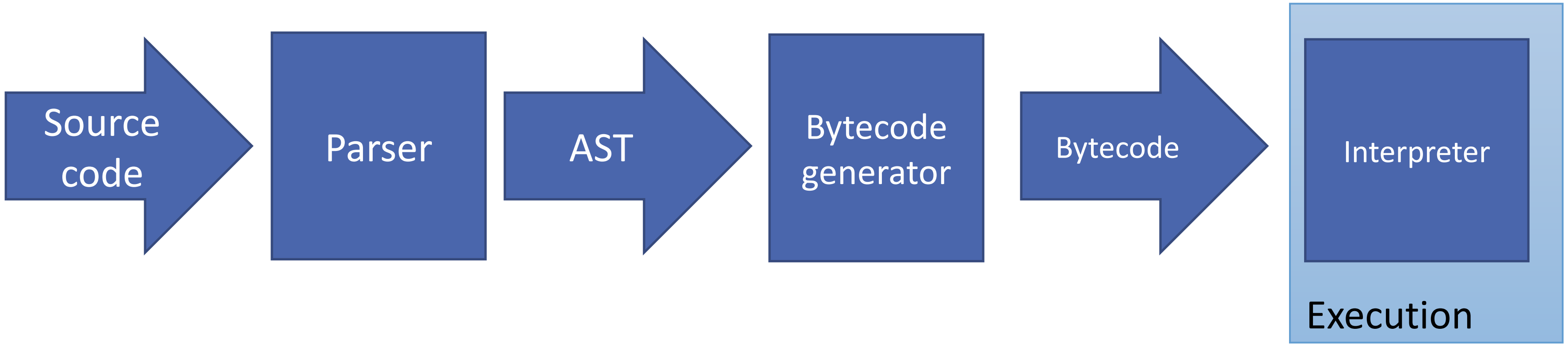
Adoption

- A non-executable stack was not immediately adopted
- The OS occasionally needed to place code in the stack
  - For example, trampoline code for handling UNIX signals
- Widely adopted today

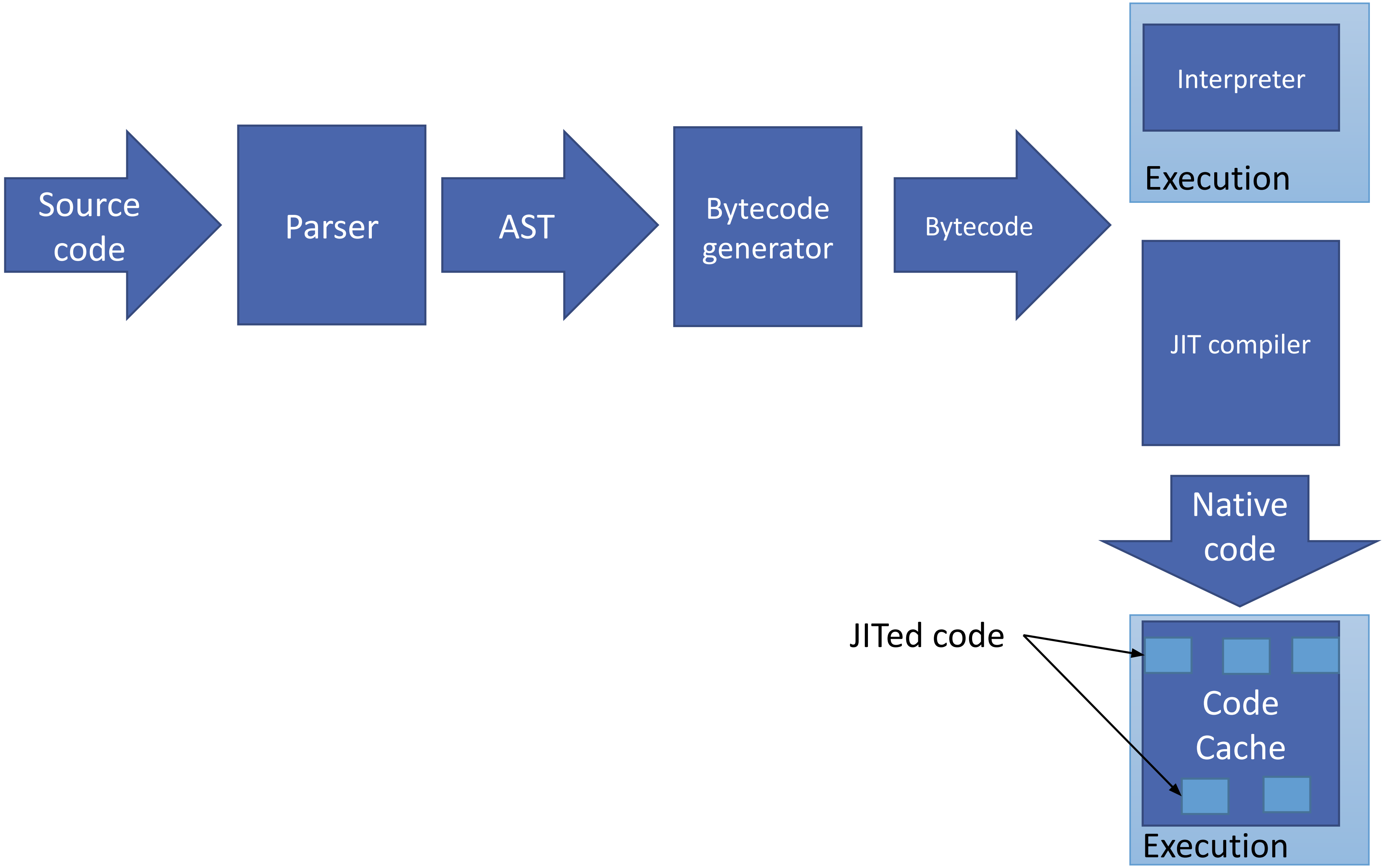
Unless You Are a Browser…

- Very popular software
  - Probably installed on every client device
- Large and complex software
- Execute JavaScript

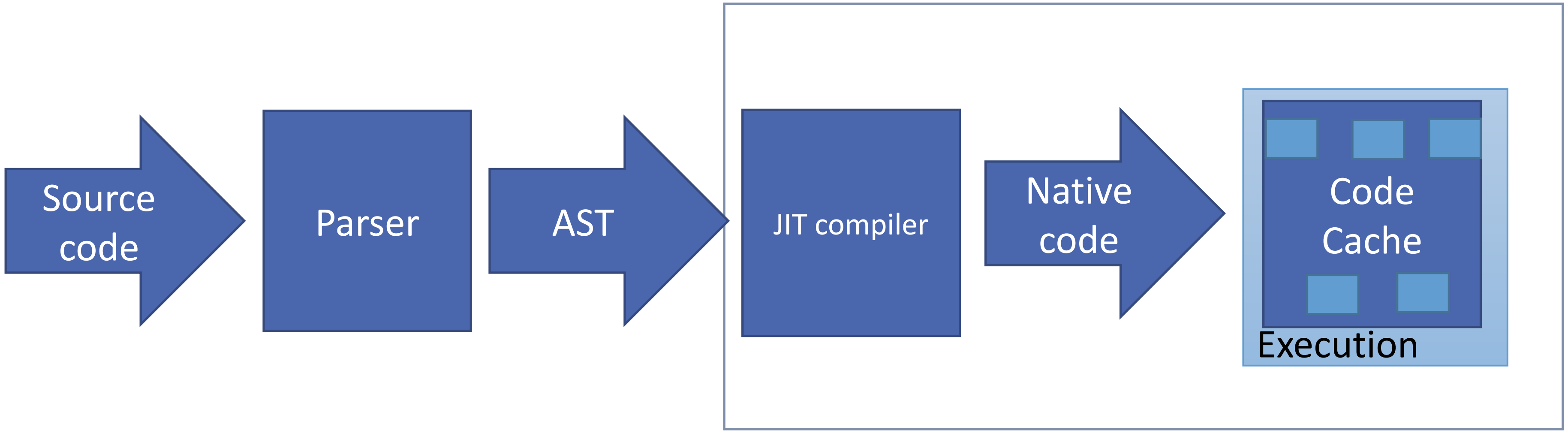
How Does JavaScript Run



How Does JavaScript Run



How Does JavaScript Run

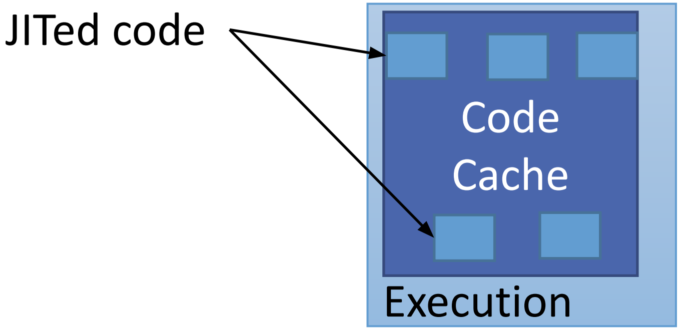


- Google V8 designed specifically to execute at speed.
- Bytecode generation skipped
- Directly emit native code
- Overall JavaScript execution improved by 150%

Code Cache

- JITed code and code cache have interesting properties from the perspective of the attacker
  - Code is continuously generated
  - Code needs to be executable

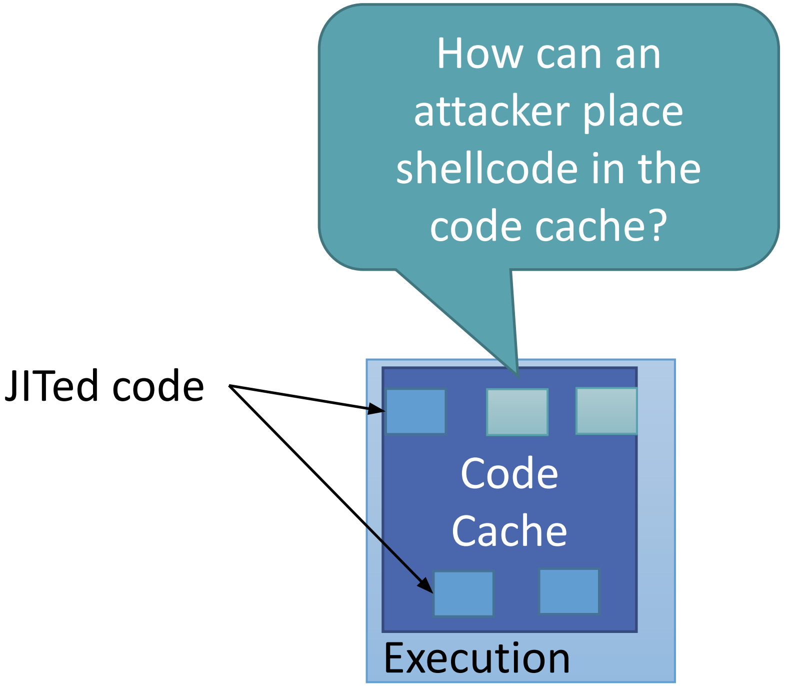
- **Violates the W^X policy**



Code Cache

- JITed code and code cache have interesting properties from the perspective of the attacker
  - Code is continuously generated
  - Code needs to be executable

- **Violates the W^X policy**



From JS to Code Cache

- JS code is JITed and placed in the code cache
- Some JS engines do not separate data and code

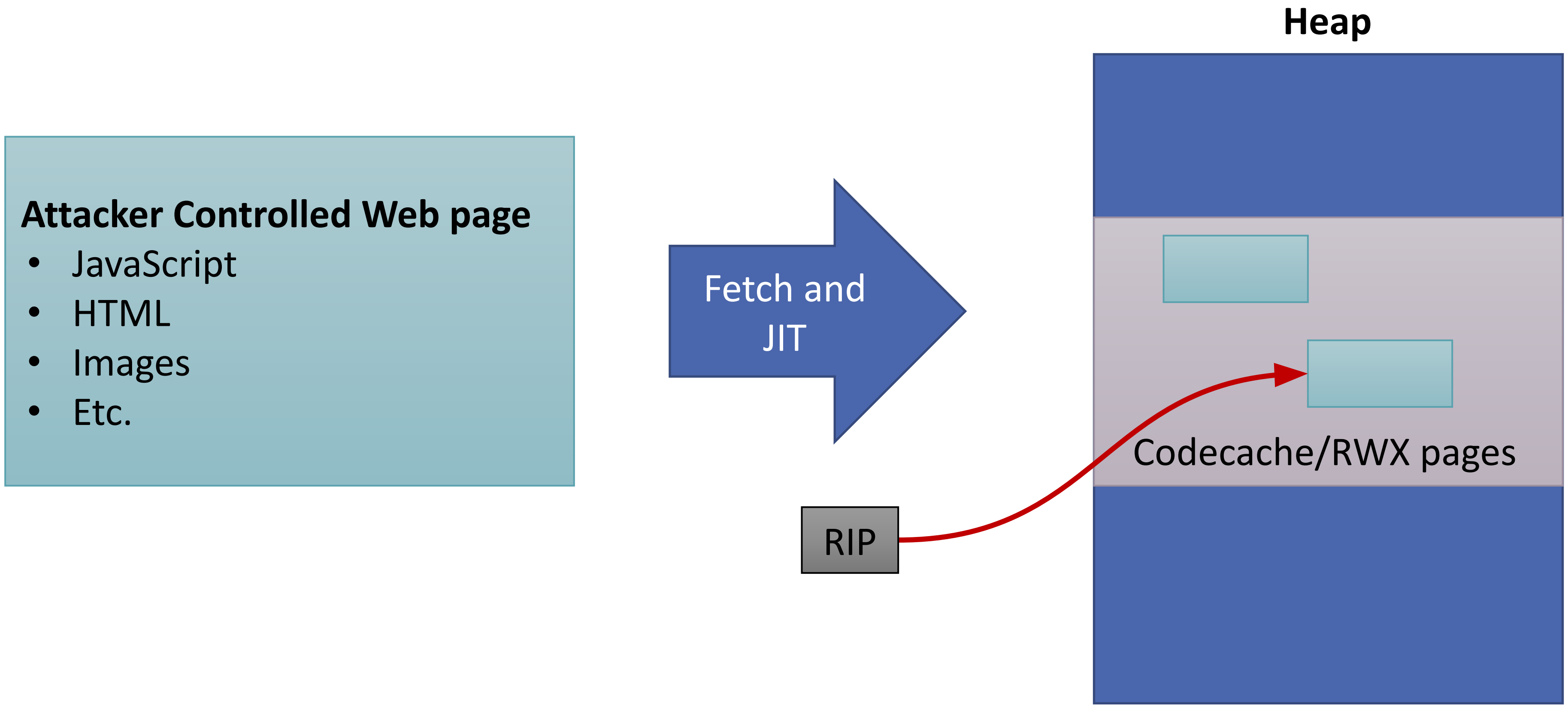
```
<html>
<body>
<script language='javascript'>

var myvar = unescape('%u\4F43%u\4552'); // CORE
myvar += unescape('%u\414C%u\214E'); // LAN!
alert("allocation done");

</script>
</body>
</html>
```

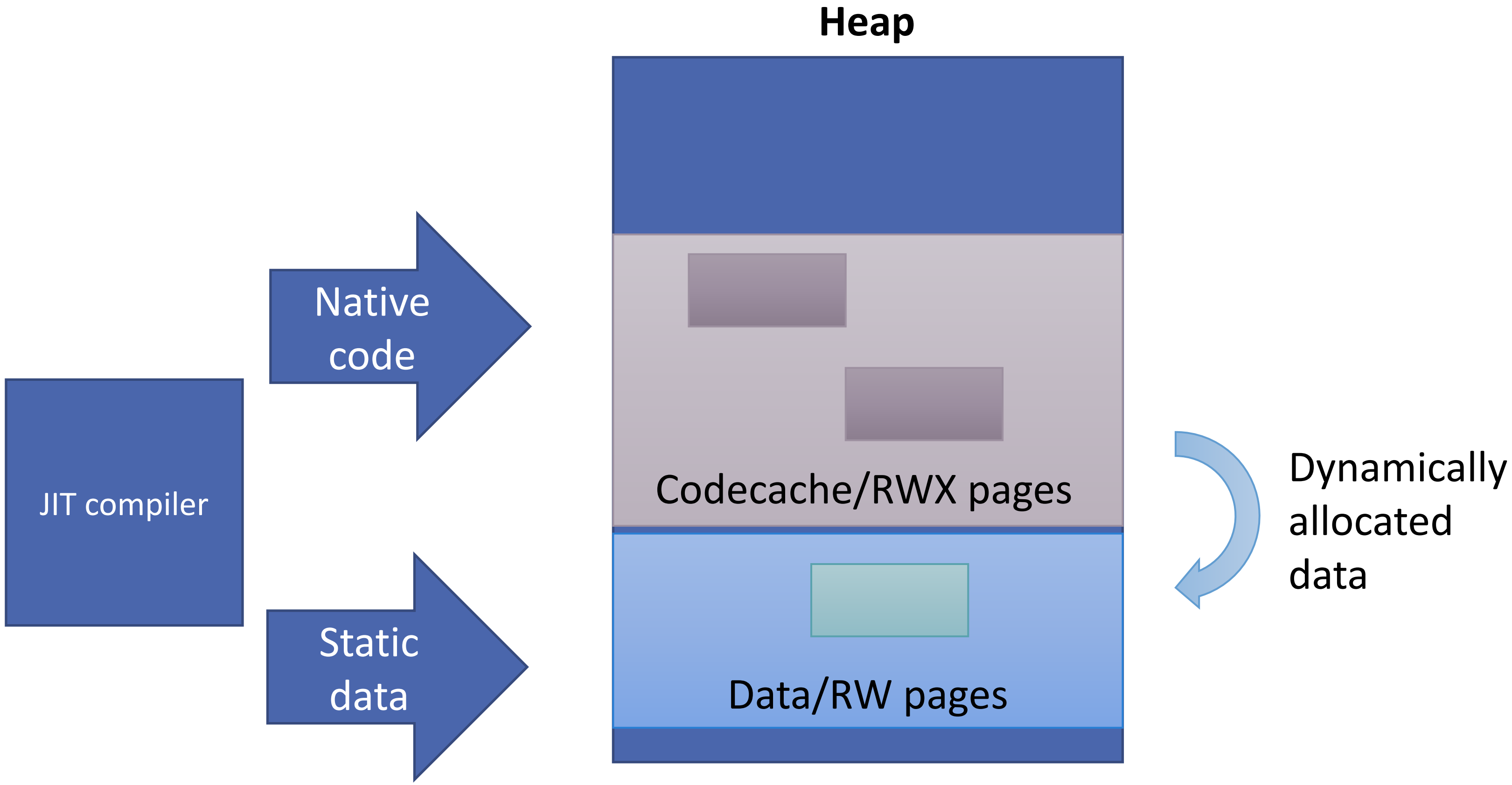
Code-Injection Attacks Against Browsers

- Return to code injected in the codecache

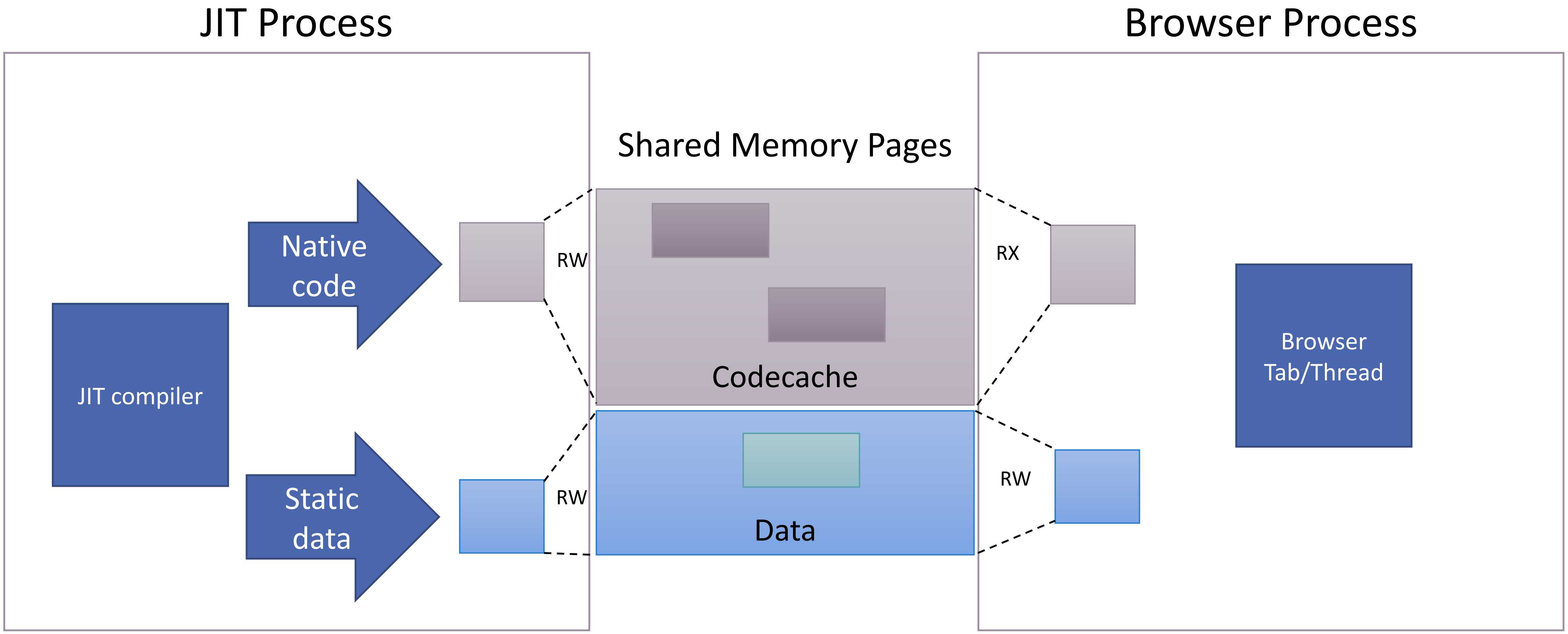


Avoiding Code Injection in Browsers

- Separate code and data into separate memory areas
- Still violates W^X



W^X Semantics in Browser Processes



Additional Reading

- The Devil is in the Constants: Bypassing Defenses in Browser JIT Engines
  - https://www.portokalidis.net/files/devilinconstants_ndss15.pdf
- libmpk: Software Abstraction for Intel Memory Protection Keys (Intel MPK)
  - https://www.usenix.org/system/files/atc19-park-soyeon.pdf