# CSci 5271
## Guest Lecture

**Navid Emamdoost**
navid@cs.umn.edu

University of Minnesota
Driven to Discover℠

# Software-based Fault Isolation

University of Minnesota
Driven to Discover℠

## Need for extensibility

University of Minnesota

- Applications can incorporate independently developed modules
  - Operating System
    - Add new file system
  - Database Management System
    - User-defined data type
  - Browser
    - Multimedia editor



## Problem with extensions

University of Minnesota

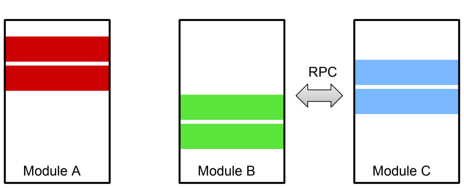
- Security and Reliability
- Extensions may be
  - Malicious
  - Vulnerable
  - Faulty
- Solution:
  - Isolate from others



## Isolation option 1

University of Minnesota

- Hardware-based isolation
  - Place each module in its own address space
  - Communicate via RPC

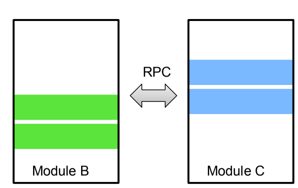





## Isolation option 1

University of Minnesota

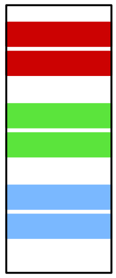
- Hardware-based isolation
  - Place each module in its own address space
  - Communicate via RPC
    - Switch to kernel mode
    - Copy arguments
    - Save/Restore registers
    - Switch address spaces
    - Return to user mode



## Isolation option 2

University of Minnesota

- Software-based isolation
  - All modules in same virtual address
  - Protect them from each other
  - Provide efficient communication

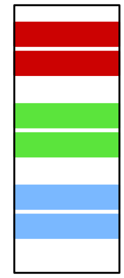




## Isolation option 2

University of Minnesota

- Software-based isolation
  - All modules in same virtual address
  - Protect them from each other
  - Provide efficient communication





# Efficient Software-based Fault Isolation

Robert Wahbe, Steven Lucco, Thomas E. Anderson, Susan L. Graham
SOSP 1993

University of Minnesota
Driven to Discover℠

## Goal

University of Minnesota

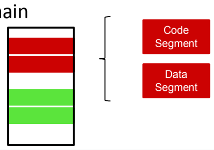
- Protect the rest of an application from a buggy/malicious module on **RISC** architecture
- Separate untrusted code
  - Define a fault domain
  - Prevent the module from jumping or writing outside of it
  - While letting efficient communications



## Fault Domain

University of Minnesota

- Load untrusted extension into its own fault domain
  - Code Segment
  - Data Segment
- Security Policy:
  - No code is executed outside of fault domain
  - No data changed outside of fault domain
    - Some protect load, too

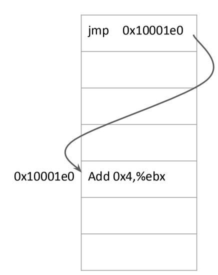





## Unsafe Instructions

University of Minnesota

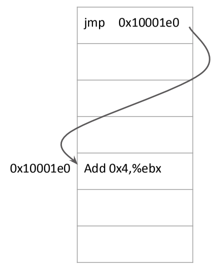
- Jump or store instructions
  - Change Control flow
  - Change data
- Addressing issue
  - jmp 0x10001e0



## Unsafe Instructions

University of Minnesota

- Jump or store instructions
  - Change Control flow
  - Change data
- Addressing issue
  - jmp 0x10001e0
  - mov %eax,0x11020028

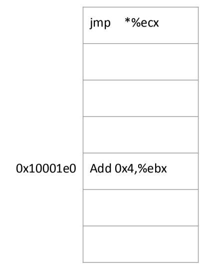





## Unsafe Instructions

University of Minnesota

- Jump or store instructions
  - Change Control flow
  - Change data
- Addressing issue
  - jmp 0x10001e0
  - jmp *%ecx
  - mov %eax,0x11020028
  - mov $0x1b80,(%ecx)






## Segment ID

University of Minnesota

- Within a segment
  - Addresses share unique pattern of upper bits


## Segment Matching

University of Minnesota

- Insert checking code before unsafe instruction
  - check segment ID of target address
- Use dedicated registers

dedicated-reg ⇐ target-address
scratch-reg ⇐ (dedicated-reg >> shift-reg)
if scratch-reg == segment-reg:
    jmp/mov using dedicated-reg

## Segment Matching

University of Minnesota

- Needs 4 dedicated registers
- Checking code must be atomic
- Exact location of fault can be detected
- Runtime overhead
  - 4 extra instructions

## Address Sandboxing

University of Minnesota

- Ensure, do not check!
- Before each unsafe instruction
  - Set upper bit of target address to correct segment ID

dedicated-reg ⇐ target-address & and-mask
dedicated-reg ⇐ dedicated-reg | segment-reg
jmp/mov using dedicated-reg

## Address Sandboxing

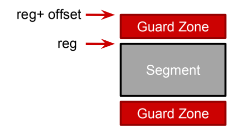


- Prevents faults
- Needs 5 dedicated registers
- 2 extra instructions
  - less overhead compared to segment matching

## Optimizations



- register-plus-offset mode
  - store value, offset(reg)
  - offset is in the range of -64K to +64K
  - mov %esi,0x8(%edx)




## Optimizations

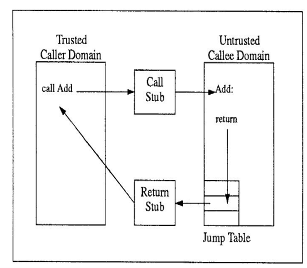


- Stack pointer
  - Just sandbox it when it is set (beginning of a function)
  - Ignore sandboxing for small changes (push, pop)
  - Works because of guard zones

## Cross Fault Domain Communication



- Host to an untrusted module
- Untrusted module to host
- Call/Return Stub
  - For each pair of fault domains






## Cross Fault Domain Call



- Trusted call/return stub
  - copy parameters
  - switch execution stack
  - maintain values of CPU registers
  - no traps or address space switching
    - faster
  - returns via jump table
    - jump targets are immediates
    - a legal address in target fault domain

## Implementation



- Change the compiler
  - emit encapsulation code into distrusted code
- At the load time
  - check the integrity of encapsulation code
  - Verifier

## Verifier



- Responsible for checking encapsulation instructions just before execution start
- Challenge:
  - indirect jump
- Hint:
  - every store/jump uses dedicated registers
- Look for changes in dedicated registers
  - any change means beginning of a check region
  - verify the integrity of check region





What about CISC architectures?!
x86



# Evaluating SFI for a CISC Architecture
(PittSFIeld)

Stephen McCamant, Greg Morrisett
USENIX 2005

University of Minnesota
Driven to Discover℠

## CISC Architectures



- RISC Architecture
  - Fixed length instructions
  - More CPU registers
- Intel IA-32 (aka x86-32)
  - Variable length instructions
  - Less CPU registers
- Classical SFI is not applicable here



## CISC Architectures



- Processor can jump to any byte
- Hard to make hidden instructions safe
- Solution: Instruction Alignment

```
         push %esi
     mov $0x56,%dh | sbb $0xff,%al | inc %eax | or %al,%dh
movzbl 0x1c(%esi),%edx |    incl 0x8(%eax)        ...
0f b6      56        1c       ff       40      08     c6
```



## Alignment

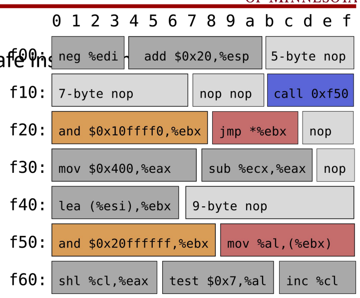


- Divide memory into 16-byte chunks
- No instruction is allowed to cross chunk boundary
- Target of jumps placed at the beginning of chunks
- Call instructions placed at the end of chunk

## Alignment

University of Minnesota

- Use *NOP* for padding
- No separation of an unsafe ins

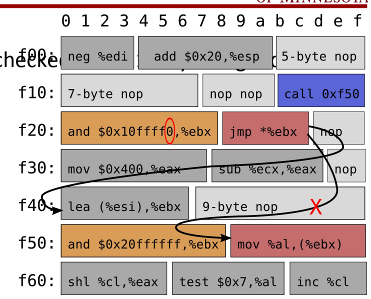





## Jumps

University of Minnesota

- Chunks are atomic
- Jump destinations are checke

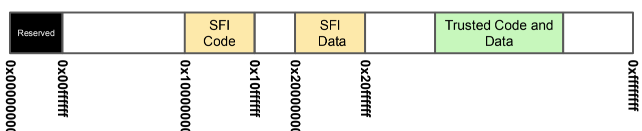





## Optimization: AND-only Sandboxing

University of Minnesota

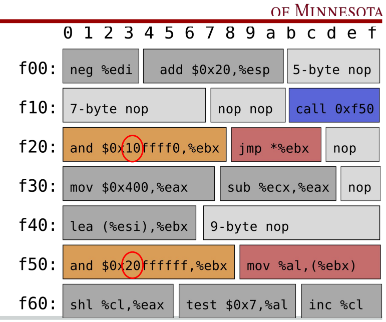
- Choose code and data region addresses carefully
- Their ID just has one bit set
- Reduces sandboxing sequence to just one instruction






## Example

University of Minnesota






## Verification

University of Minnesota

- Statically check
  - No jump to outside of code region
  - No store to outside of data region
- Before each unsafe jump or store there should be a sandboxing AND
- The sandboxing AND should not be the last instruction in a chunk



## Performance overhead

University of Minnesota

- Implemented prototype
  - named PittSFIeld
- Average module overhead: 21%
- But the overall execution can be improved because of faster communications
  - no trap, RPC, etc

## Native-client: A Sandbox for Portable, Untrusted x86 Native Code

Bennet Yee, et al.
IEEE S&P, 2009

University of Minnesota
Driven to Discover℠

## Google Native Client

University of Minnesota

- Browser Plugin (Google Chrome)
  - Allows execution of untrusted C/C++ code in browser
- Browser?! Native Code?!
  - Yes! browsers are new platform for applications
- Gives Browser plugins performance of native code
- Ships by default since Chrome 14



## Sandboxing

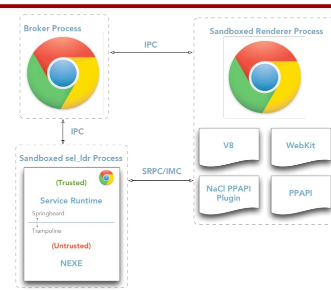
University of Minnesota

- Inner Sandbox
  - Code sandboxing
    - Alignment and address sandboxing
    - Check branch target addresses
  - Data Sandboxing
    - segmented addressing mode supported by x86_32
- Outer Sandbox
  - Controls system calls issued by native code
  - Whitelist



## NaCl Architecture

University of Minnesota




Source: https://media.blackhat.com/bh-us-12/Briefings/Rohlf/BH_US_12_Rohlf_Google_Native_Client_Slides.pdf



## Inner Sandbox

University of Minnesota

- On x86_32
  - Sandboxing via segmented memory
  - Used to separate trusted from untrusted code/data
  - Modified when switching between trusted/untrusted
  - `%cs` code
  - `%ds` data
  - `%gs` thread local storage
  - `%ss %es %fs` all set to `%ds`
- On x86_64
  - mov/branch alignment, guard pages
  - `r15` keeps base address of an aligned 4GB range



## Native Client Toolchain

University of Minnesota

- Modified GCC and GAS
  - To emit sandboxing instructions
- Final executable has ELF file structure (called NEXE)
  - Can be disassembled using standard tools
  - objdump -d

```
naclcall   %ebx   ---->   and  $0xffffffe0,%ebx
                          call      *%ebx

nacljmp  %ecx     ---->   and  $0xffffffe0,%ecx
                          jmp  *%ecx
```

# Alignment

UNIVERSITY OF MINNESOTA

- Divide memory into 32-byte Bundles
- Target of jumps placed at the beginning of bundles
- No instruction is allowed to cross bundle boundary

```
                              ...
1060451:    83 c8 01                 or     $0x1,%eax
1060454:    88 41 28                 mov    %al,0x28(%ecx)
1060457:    8b 0e                    mov    (%esi),%ecx
1060459:    8b 51 04                 mov    0x4(%ecx),%edx
106045c:    85 d2                    test   %edx,%edx
106045e:    66 90                    xchg   %ax,%ax

1060460:    0f 88 9a 01 00 00        js     1060600
1060466:    8b 01                    mov    (%ecx),%eax
1060468:    8d 3c 95 00 00 00 00     lea    0x0(,%edx,4),%edi
106046f:    89 d6                    mov    %edx,%esi
1060471:    83 ea 01                 sub    $0x1,%edx
1060474:    83 e6 07                 and    $0x7,%esi
1060477:    83 fa ff                 cmp    $0xffffffff,%edx
106047a:    8d b6 00 00 00 00        lea    0x0(%esi),%esi

1060480:    c7 04 38 84 06 03 11     movl   $0x11030684,(%eax,%edi,1)
1060487:    8d 47 fc                 lea    -0x4(%edi),%eax
106048a:    0f 84 70 01 00 00        je     1060600
                              ...
```



# Alignment

UNIVERSITY OF MINNESOTA

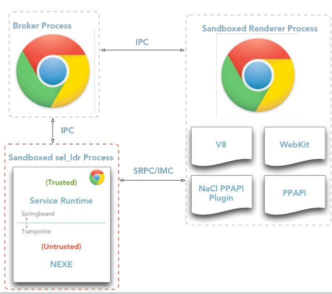
- Divide memory into 32-byte Bundles
- Call instructions placed at the end of bundles
  - For the sake of return address alignment

```
10ee700:    55                       push   %ebp
10ee701:    89 e5                    mov    %esp,%ebp
10ee703:    83 ec 18                 sub    $0x18,%esp
10ee706:    c7 04 24 2c 5e 01 11     movl   $0x11015e2c,(%esp)
10ee70d:    8d b4 26 00 00 00 00     lea    0x0(%esi,%eiz,1),%esi
10ee714:    8d bc 27 00 00 00 00     lea    0x0(%edi,%eiz,1),%edi
10ee71b:    e8 a0 72 f5 ff           call   10459c0

10ee720:    89 ec                    mov    %ebp,%esp
10ee722:    5d                       pop    %ebp
10ee723:    59                       pop    %ecx
10ee724:    83 e1 e0                 and    $0xffffffe0,%ecx
10ee727:    ff e1                    jmp    *%ecx
                              ...
```



# Service Runtime

UNIVERSITY OF MINNESOTA

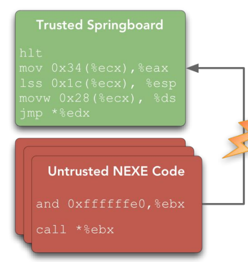



Source: https://media.blackhat.com/bh-us-12/Briefings/Rohlf/BH_US_12_Rohlf_Google_Native_Client_Slides.pdf



# Service Runtime

UNIVERSITY OF MINNESOTA




The untrusted NEXE code is unable to transfer control to an arbitrary location in the trusted Springboard due to the alignment mask before the call. It may only execute the first instruction, which is a *hlt*

Source: https://media.blackhat.com/bh-us-12/Briefings/Rohlf/BH_US_12_Rohlf_Google_Native_Client_Slides.pdf



# Service Runtime

UNIVERSITY OF MINNESOTA

- No existing or new memory allocations may be marked as executable at runtime
  - This guarantees only validated code pages have executable permissions
- NaCl syscall
  - ~ 30 syscalls allowed
  - Basic operations such as open, close, read, write, ioctl, mmap, munmap, stat, _exit and a few others



# Validator

UNIVERSITY OF MINNESOTA

- Disassembles all NEXE instructions
- Starts at trusted 32 byte aligned entry point
- Exits on any blacklisted instructions
  - Privileged instructions
  - Instructions that modify segment registers
  - ret
  - sysenter

# CBI NaCl

or
Cross-Bundle Instruction Native Client

University of Minnesota
Driven to Discover℠



## Padding vs Performance

University of Minnesota

- Change NaCl padding scheme
  - Pad removal
  - Greedy Algorithm
- Multipass Validator
  - We must guarantee sandboxing policy enforcement
  - Appropriate changes in validator



## Types of Padding

University of Minnesota

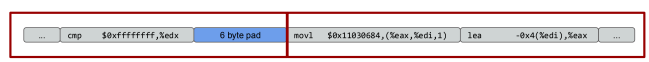
- Indirect jump target
  - Will be placed at the next bundle start
- Call instruction
  - Will be placed at the end of the bundle
- Cross bundle instruction
  - Will be pushed to the start of next bundle



## Types of Padding

University of Minnesota

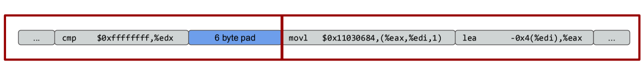
- Indirect jump target
  - Will be placed at the next bundle start
- Call instruction
  - Will be placed at the end of the bundle
- Cross bundle instruction
  - Will be pushed to the start of next bundle
  - Conservative






## Pad Removal

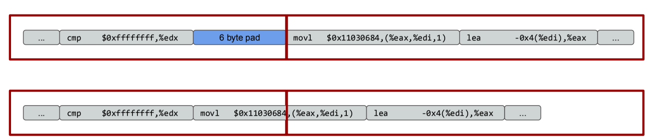
University of Minnesota





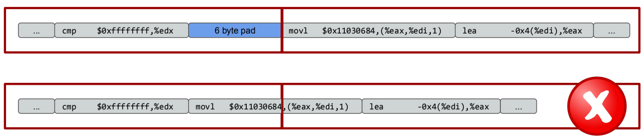

## Pad Removal

University of Minnesota



## Pad Removal

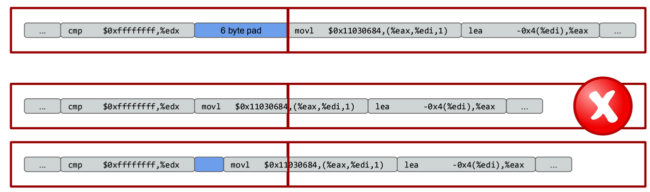







## Pad Removal







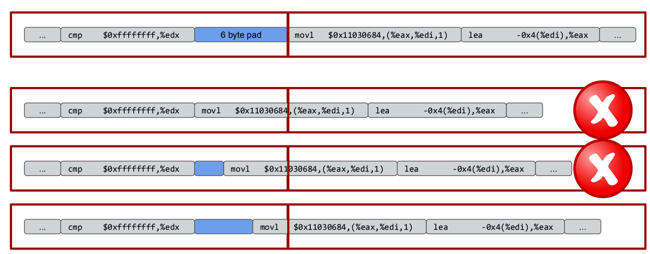

## Pad Removal






## Pad Removal

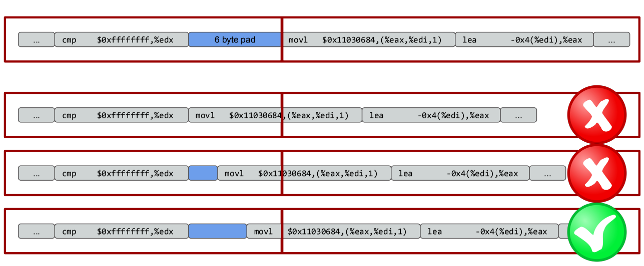







## Pad Removal








## NaCl Validator



- One pass: from the start to the end of code
- Maintains two bitmaps: *valid* and *target*
- At each address checks the instruction
- If a valid instruction marks it in *valid* and advance by instruction size
- If indirect branch checks masking instruction presence
- If direct branch, the destination is marked in *target*
- At the end *target* and *valid* are compared together

## Multipass Validator

University of Minnesota

- Challenge: Cross-Bundle Instructions

```
                                   ...
1060451:     83 c8 01                 or     $0x1,%eax
1060454:     88 41 28                 mov    %al,0x28(%ecx)
1060457:     8b 0e                    mov    (%esi),%ecx
1060459:     8b 51 04                 mov    0x4(%ecx),%edx
106045c:     85 d2                    test   %edx,%edx
106045e:     66 90                    xchg   %ax,%ax

1060460:     0f 88 9a 01 00 00        js     1060600
1060466:     8b 01                    mov    (%ecx),%eax
1060468:     8d 3c 95 00 00 00 00     lea    0x0(,%edx,4),%edi
106046f:     89 d6                    mov    %edx,%esi
1060471:     83 ea 01                 sub    $0x1,%edx
1060474:     83 e6 07                 and    $0x7,%esi
1060477:     83 fa ff                 cmp    $0xffffffff,%edx
106047a:     8d b6 00 00 00 00        lea    0x0(%esi),%esi

1060480:     c7 04 38 84 06 03 11     movl
$0x11030684,(%eax,%edi,1)
1060487:     8d 47 fc                 lea    -0x4(%edi),%eax
106048a:     0f 84 70 01 00 00        je     1060600
                                   ...
```



## Multipass Validator

University of Minnesota

- Challenge: Cross-Bundle Instructions
- Multipass: start validation from every crossing point
  - Bundle start
  - Stop if reached to an already validated instruction

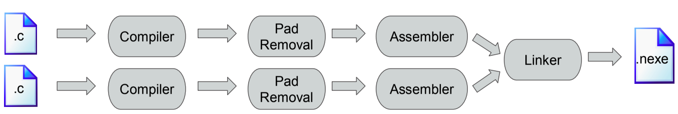
```
                                   ...
1060451:     83 c8 01                 or     $0x1,%eax
1060454:     88 41 28                 mov    %al,0x28(%ecx)
1060457:     8b 0e                    mov    (%esi),%ecx
1060459:     8b 51 04                 mov    0x4(%ecx),%edx
106045c:     85 d2                    test   %edx,%edx
106045e:     66 90                    xchg   %ax,%ax

1060460:     0f 88 9a 01 00 00        js     1060600
1060466:     8b 01                    mov    (%ecx),%eax
1060468:     8d 3c 95 00 00 00 00     lea    0x0(,%edx,4),%edi
106046f:     89 d6                    mov    %edx,%esi
1060471:     83 ea 01                 sub    $0x1,%edx
1060474:     83 e6 07                 and    $0x7,%esi
1060477:     83 fa ff                 cmp    $0xffffffff,%edx
106047a:     8d b6 00 00 00 00        lea    0x0(%esi),%esi

1060480:     c7 04 38 84 06 03 11     movl
$0x11030684,(%eax,%edi,1)
1060487:     8d 47 fc                 lea    -0x4(%edi),%eax
106048a:     0f 84 70 01 00 00        je     1060600
                                   ...
```



## Separate Compilation

University of Minnesota

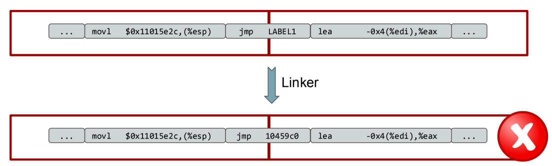
- We process each source file separately
  - Decide about the paddings to be removed
  - Assemble them into object files (using modified *GAS*)
  - Then link them together






## Relocations Problem

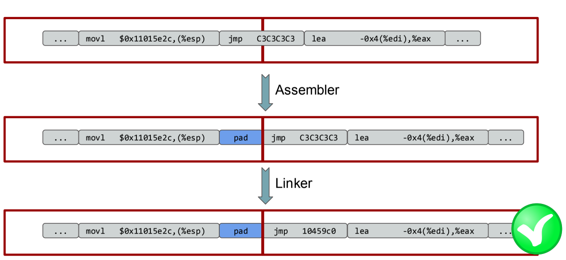
University of Minnesota






## Relocations Problem

University of Minnesota






University of Minnesota

Thank you

Any Question?