Identifying and Analyzing Pointer Misuses for Sophisticated Memory-corruption Exploit Diagnosis

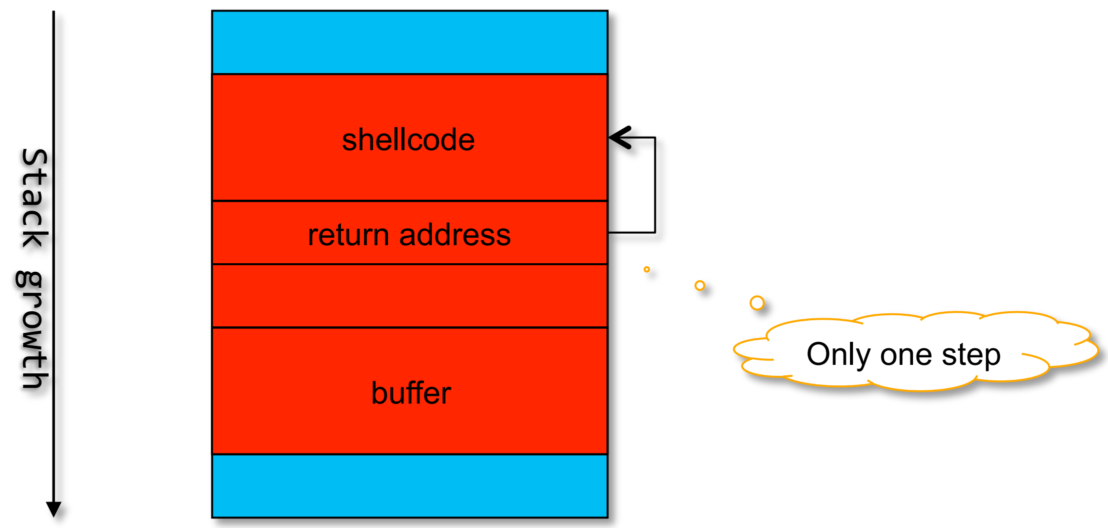
Mingwei Zhang (†)
Aravind Prakash (§)
Xiaolei Li (†)
Zhenkai Liang (†)
Heng Yin (§)

(†) School of Computing, National University of Singapore

(§) Department of Electrical Engineering and Computer Science, Syracuse University

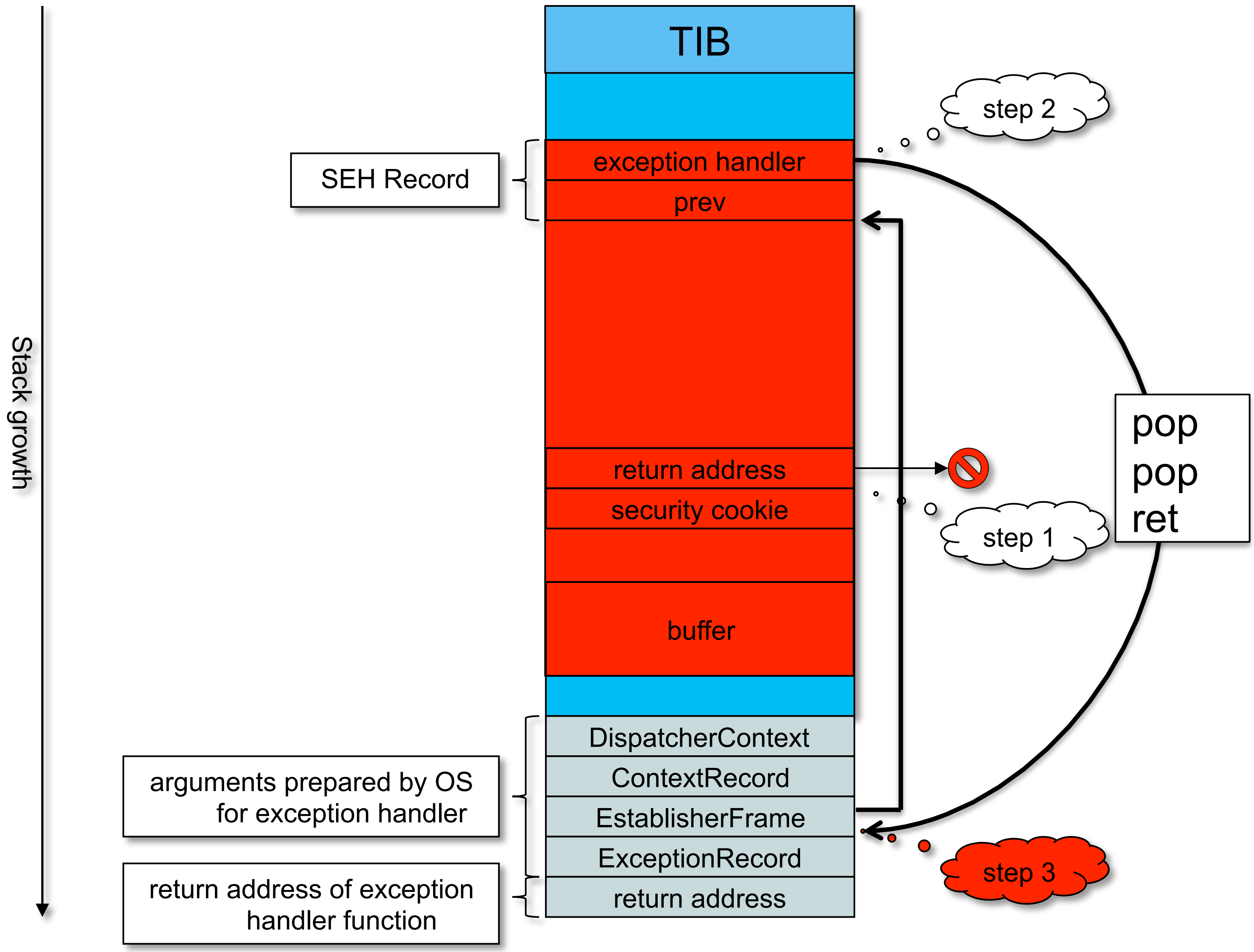
Simple Stack Buffer Overflow

- An attacker overwrites vulnerable function return address, which points to shellcode on stack.



- These single step attacks don't work anymore thanks to:
  - ASLR, DEP, NX, etc.

Exploiting SEH Mechanism



Insights

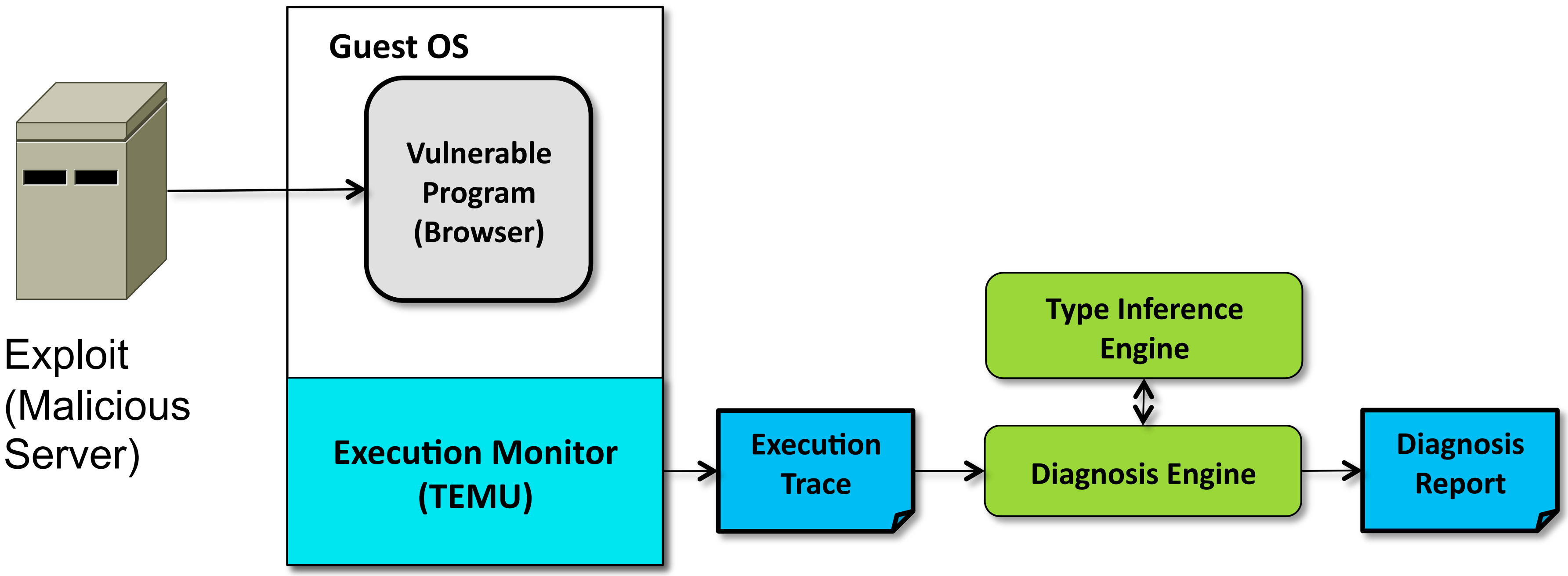
- Recent attacks employ **multiple steps**.
- **Pointer misuse** is very prominent in sophisticated attacks.
- Key steps constitute pointer misuses.

Our Goal:
Diagnosing pointer misuses in a multi-step attack.

Pointerscope – Attack Diagnosis Engine

- **Type System** tailored to diagnose pointer misuse.
- **Eager type inference** system to detect pointer misuses.
- Provide **big picture** of the misuse through *key steps* graph.

Overview

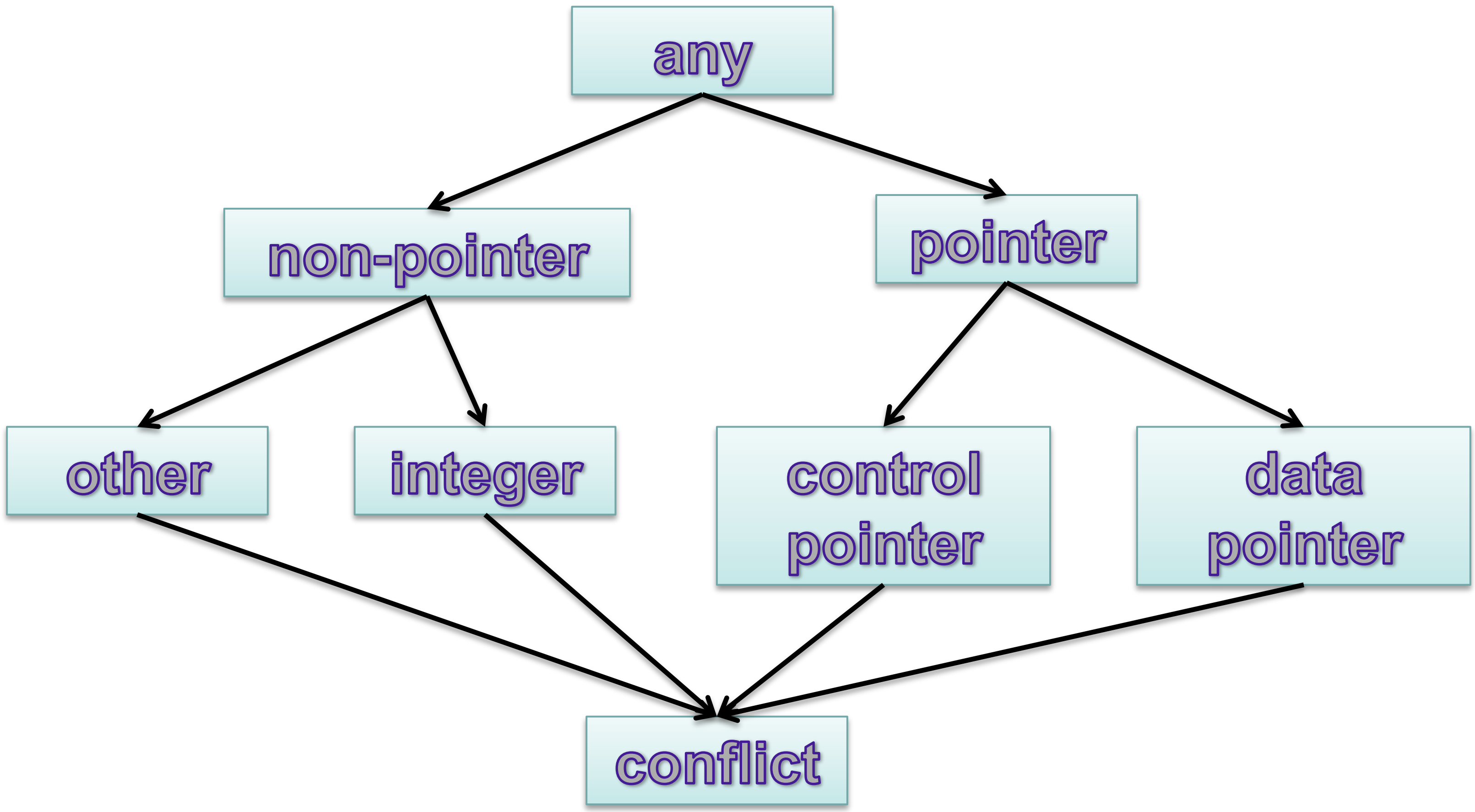


Exploit
(Malicious
Server)
Guest OS
Vulnerable
Program
(Browser)
Execution Monitor
(TEMU)
Execution
Trace
Type Inference
Engine
Diagnosis Engine
Diagnosis
Report

Variable And Variable Type

- A variable is a memory location or a register.
- Simple primitive variable types:
  - Integer
  - Control Pointer (or code pointer)
  - Data Pointer
  - Other. (The rest of the types)

Type Lattice



Eager Dynamic Type Inference

- Type Propagation:
  - mov %eax, %ebx
    - Inference: eax and ebx have same type
- Type Constraints:
  - call %eax
    - Inference: %eax contains Control Pointer

Example – Type Inference

{eax, ebx} : **ANY**	mov %eax, %ebx
{eax, ebx, ecx} : **ANY**	mov %ebx, %ecx
ecx is an INT {eax, ebx, ecx}: **Integer**	imul $0x05, %ecx, %ecx
{eax, ebx, ecx, edx} : **Integer**	mov %ecx, %edx
Used as a pointer. **Conflict**	call *%ecx

Harder than it seems!

Challenges

- X86 supports base-index with displacement – Problem: Compilers don't follow convention.

```
movl $0x8(%eax,%edx), %ecx
```

address = displacement + base register + index register

Solution: Register closest to result is the base.

Challenges… contd.

- Individual instructions not always lead to accurate type inferences.
  - Eg:

```
not %ebp
or  $0x3,%ebp
not %ebp
```

⇔

```
and $0xfffffffc,%ebp
```

  - Solution: recognizing the common patterns and treat them as special cases

Challenges… contd.

- LEA designed to load effective address, but often used in arithmetic.

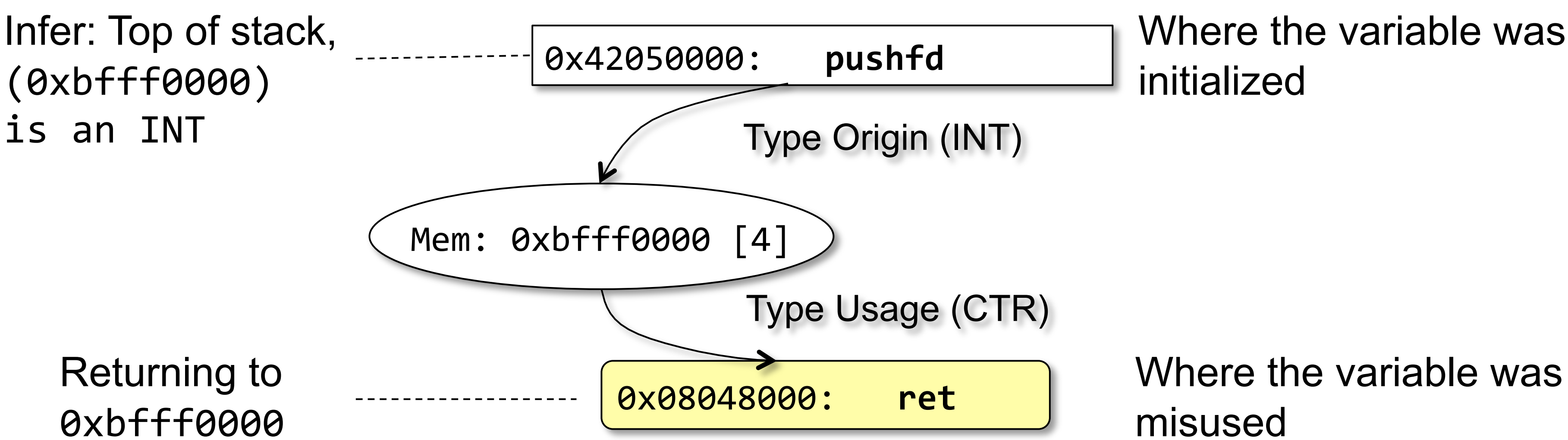
lea $0x8(%eax,%edx,4), %ecx	⇨	%ecx＝%eax＋%edx×4＋$0x8

Solution: Treat lea as an arithmetic operation.

Challenges…

- More challenges discussed in the paper!

Key Steps Graph – Example



Evaluation

- Implementation
  - Execution monitor on TEMU.
  - 3.6K lines of C code.
- Experimental setup
  - Evaluated against real world exploits from Metasploit framework.

Summary of Effectiveness

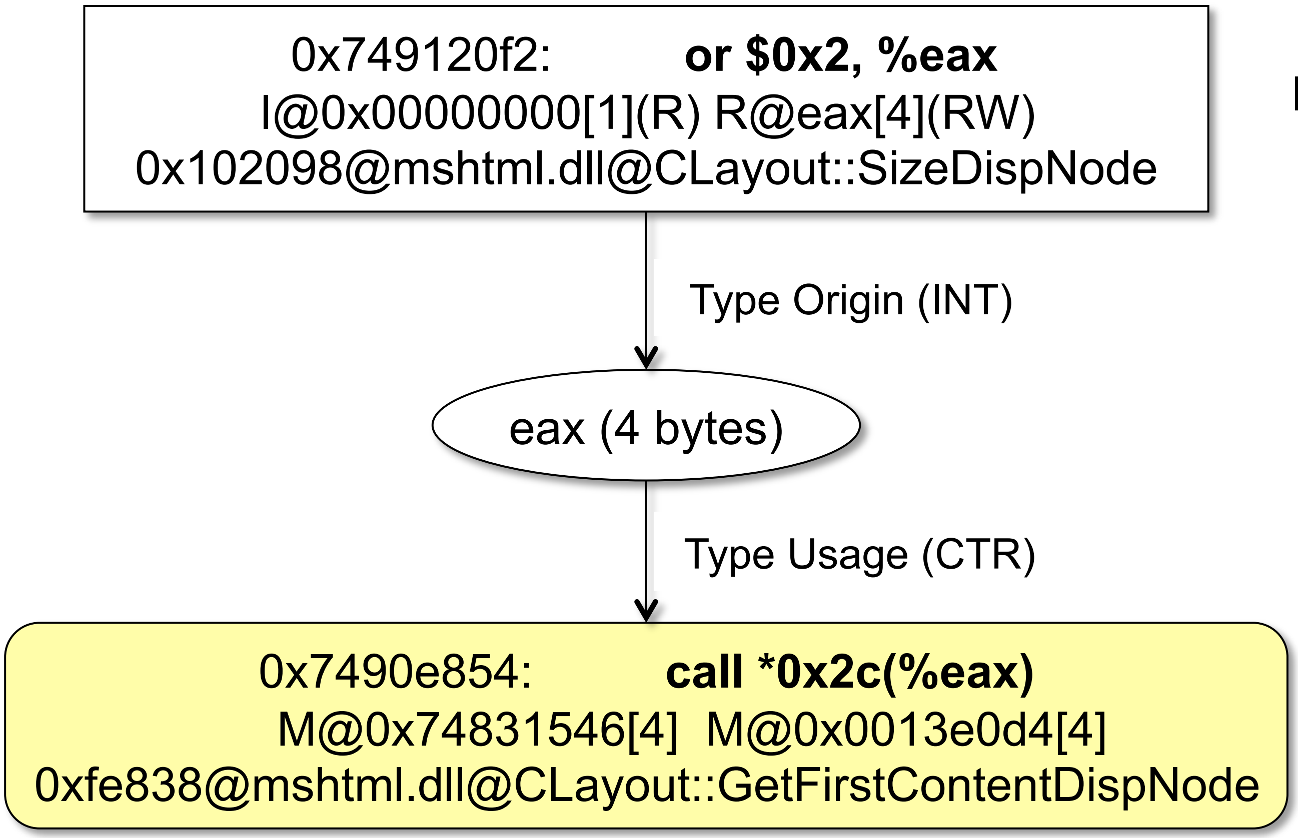
CVE	Attack Technique	Runtime*	Pointer Misuses	Trace Size	Slice Size
CVE-2010-0249	Uninitialized memory; heap spray	18m23s, 8m30s	11	307,987,560	48,404,242
CVE-2009-3672	Incorrect variable initialization; heap-spray	3m10s, 31s	2	22,759,299	955,325
CVE-2009-0075	Uninitialized memory; heap spray	25m, 21m16s	6	411,323,083	44,792,770
CVE-2006-0295	Heap overflow; heap spray	3m5s, 1s	3	808,392	34,883
CVE-2006-1016	Stack overflow; SEH exploit	4m59s, 1m33s	3	64,355,691	1,334,253
CVE-2006-4777	Integer overflow; heap spray	1m45s, 40s	3	2,632,241	1,669,751
CVE-2006-1359	Incorrect variable initialization; heap spray	11m58s, 13s	2	8,336,193	29,520
CVE-2010-3333	Stack overflow vulnerability; SEH exploit	18m53s, 7m24s	1	236,331,307	814,305
CVE-2010-3962	Incorrect variable initialization; heap spray	10m36, 15s	2	9,281,019	78,704

*Time taken to generate trace, time taken to generate key steps

Case Study: CVE-2009-3672

- This is a real world exploit for vulnerable version of IE Browser
- This attack is caused by a vulnerability in the class `CDispNode`'s member function `SetExpandedClipRect`

The First Type Conflict

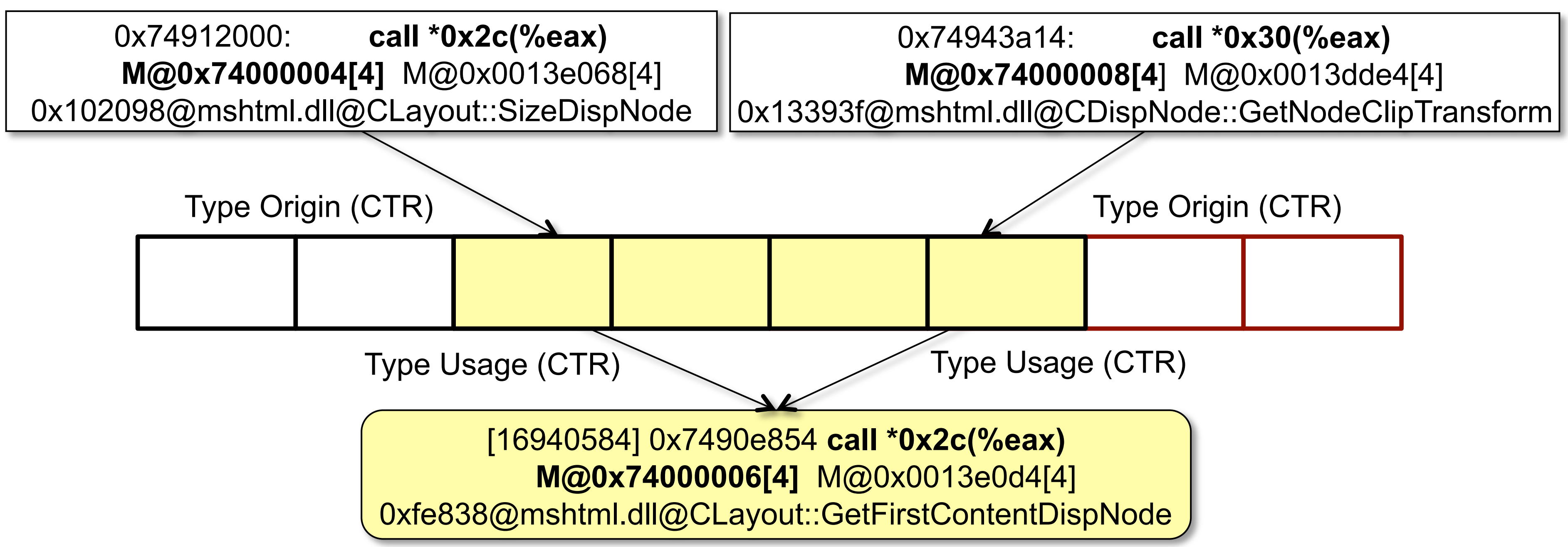


Infer: Integer

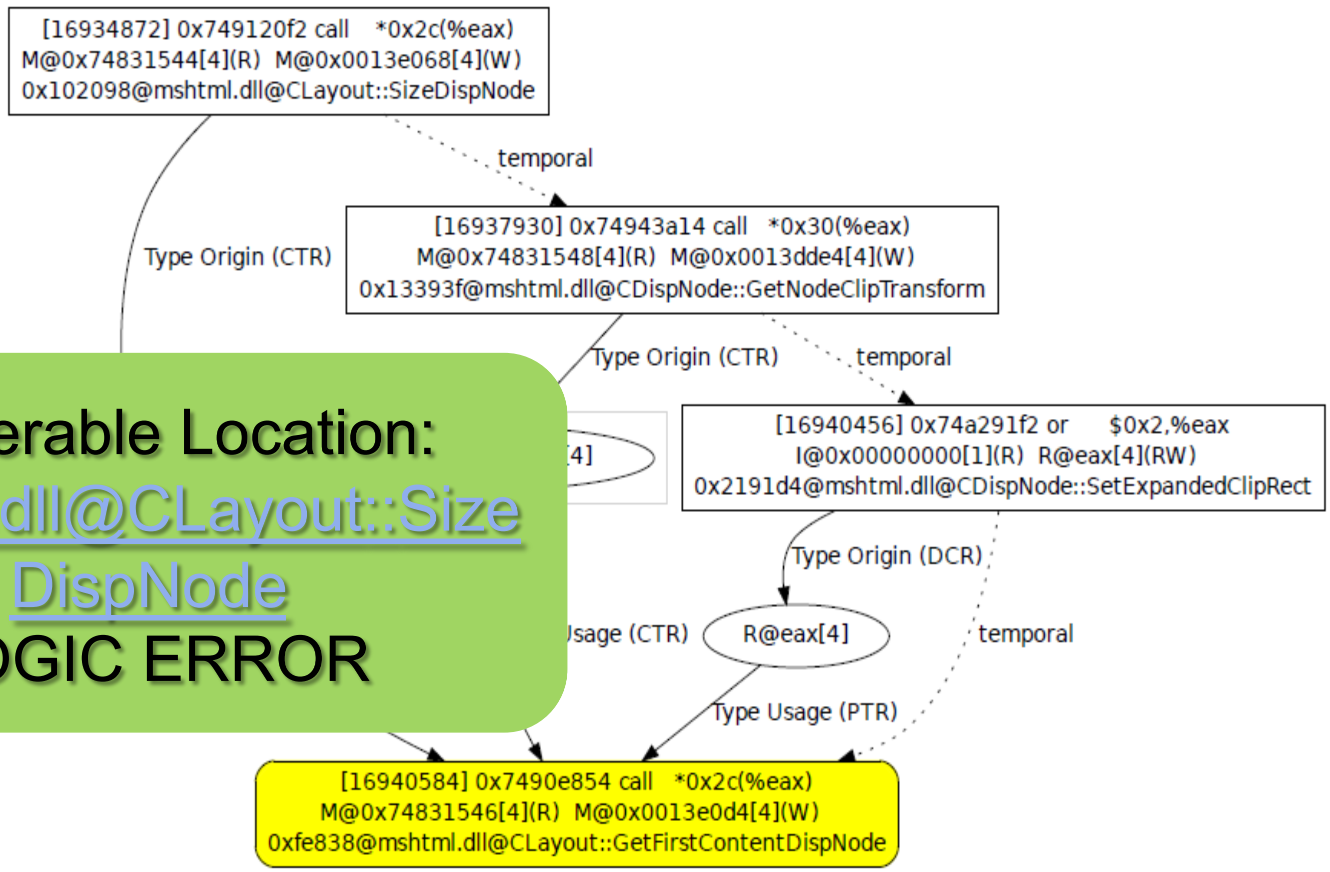
or $0x2, %eax

Used as Control Ptr Violation

The Second Type Conflict



Final Result



Vulnerable Location:
mshtml.dll@CLayout::SizeDispNode
LOGIC ERROR

Reducing False Positives

- What makes it hard?
  - Compiler optimizations
  - Code obfuscation – even by proprietary code.

- Note: Our goal is NOT to eliminate False Positives.

Related Work

- Attack Diagnosis Techniques
  - BackTracker [King, et. al, SOSP'03], Dynamic Taint Analysis [Newsome, et. al, NDSS'05]
- Type and Data Structure recovery from binary
  - Rewards [Zin, et.al, NDSS'10], Howard [Slowinska, NDSS'11], Tie [Lee, et.al, NDSS'11]
- Defense and evasion techniques
  - CFI [Abadi, et.al, CCS'05], DFI [Castro, et.al, OSDI'06], WIT [Akritidis, et.al, IEEE S&P'08]

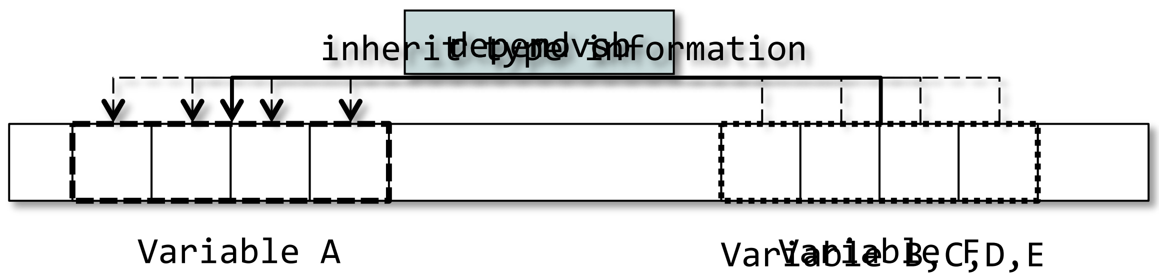
Conclusion

- We define a **pointer centric type system** to track pointers.
- We design a **type inference system** to detect pointer misuses.
- We **generate the key steps graph** to identify key steps.
- We evaluate our work by testing our system on real-world exploits from metasploit.

Questions?

Challenges… contd.

- Handling memory copy operations
  - Memory copy operations may break the integrity of variables



Solution: Aggregation.

Case Study: SEH Attack