UnROP

Scan RoP chains in a process memory dump

Kang Li **kangli@cs.uga.edu**
Xiaoning Li **ldpatchguard@gmail.com**
Lee Harrison **lee2704@uga.edu**

About us

- Kang
  - College Educator
- Xiaoning
  - Security Researcher
- Lee
  - Code Warrior

A Brief History of Exploitation Development

- Exploitation development is like performing a surgery
  - "dissect" the binary corpus

  - inject or stitch things together

Image source: http://www.shutterstock.com/pic-69407347/

Ancient Time Exploitation Dev

Code injection prior to DEP

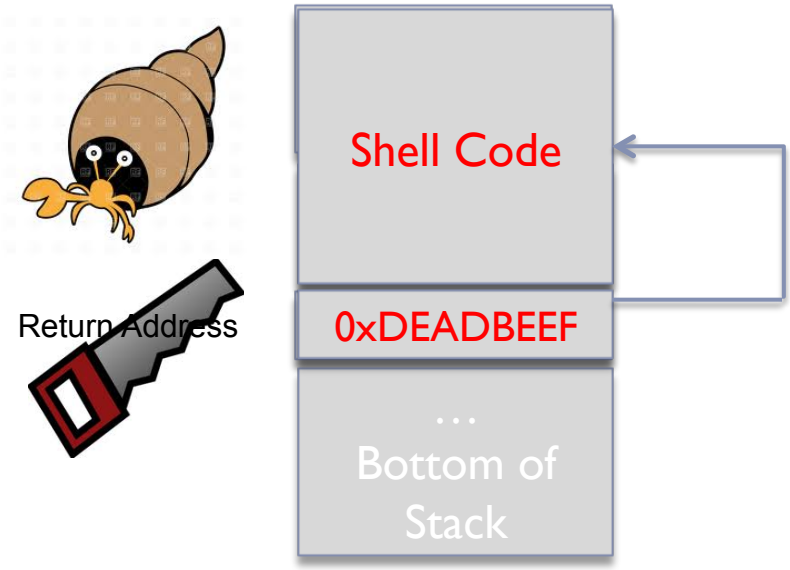


Image Source: http://www.toonpool.com/user/997/files/cave_fire_drills_895405.jpg
http://rfclipart.com/crab-in-seashell-1450-vector-clipart.html

Evolution of Exploitation Development

- Writing exploits is never easy. But compared to modern day, the development in the past was "rough".

http://www.shutterstock.com/pic-69407347/

http://toonclips.com/design/11941

DEP

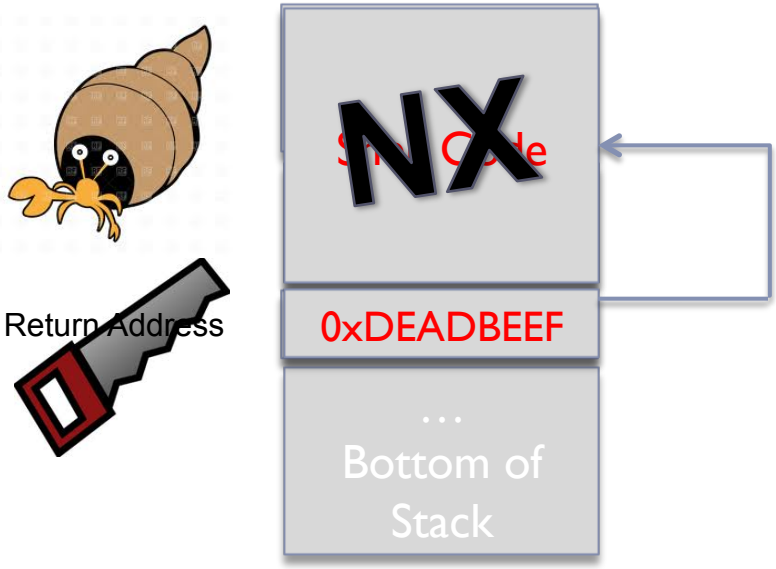


Image Source: http://www.toonpool.com/user/997/files/cave_fire_drills_895405.jpg

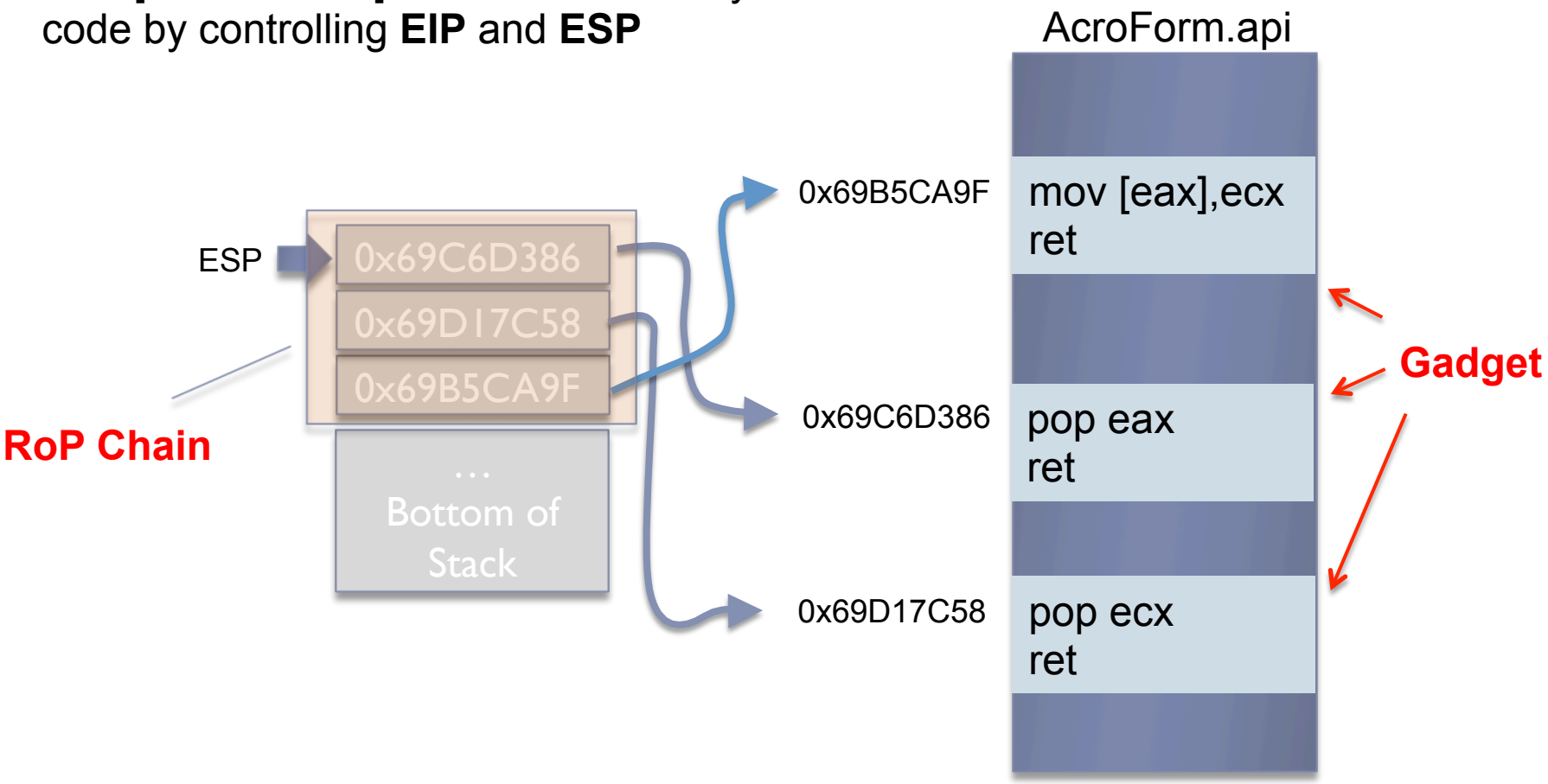
Bypassing DEP

- Ret2libc [Solar Designer 1997]
- Ret2libc chaining [Nergal 2001]
- RoP [Shacham 2007]
- Practical RoP [Dino Dai Zovi 2010]

- and many other work …

ROP Gadget Chain Example

RoP [Shacham'07] -- Execute arbitrary code by controlling **EIP** and **ESP**



Modern Day Exploitation

- Multiple stages
  - Making the cut
    - Use the vulnerability
  - Bypassing DEP
    - ROP to setup Exec Env
      - E.g. call VirtualAlloc, VirtualProtect
    - assume ASLR being handled …
  - Executing shellcode

ww.clipartpal.com/clipart/science/surgeon_148938.html

http://rfclipart.com/crab-in-seashell-1450-vector-clipart.html

Observations

1. RoP is a popular prelude of new exploitations.

2. RoP chain is often used right after the control flow change.

From the Malware Analysis Point of View

- Tracing back to the vulnerability requires
  - Fire-up VMs and load malware samples
  - Post exploitation detection
    - Such as detecting shellcode
  - Backtrace the gadgets chain
    - The step before the first gadget is usually the vulnerability.
  - Manually inspect a small RoP chain might be "enjoyable".

Picture from http://www.clker.com/cliparts

Some RoP Chains can be Complex

http://www.echometal.ca/products%20pics/boom%20chain%201.JPG

Complex RoP Chain in the Real World

- Some exploitations use massive amount of gadgets
  - PDF CVE 2013-0640
- Long chains are generated by Gadget Compiler
  - Many tools are available
    - RopMe, RopGadget, OptiROP
    - ROPC: convert code to gadget chain
      - https://github.com/pakt/ropc
  - Expect to be more common in the future.

Need a RoP Discovery and "deCompile" Tool

unRoP Tool

- Input:
  - Full process memory dump
    - code, heap, stack, shared library …
- Action:
  - Search for RoP chains in memory
- Output:
  - RoP chain candidates

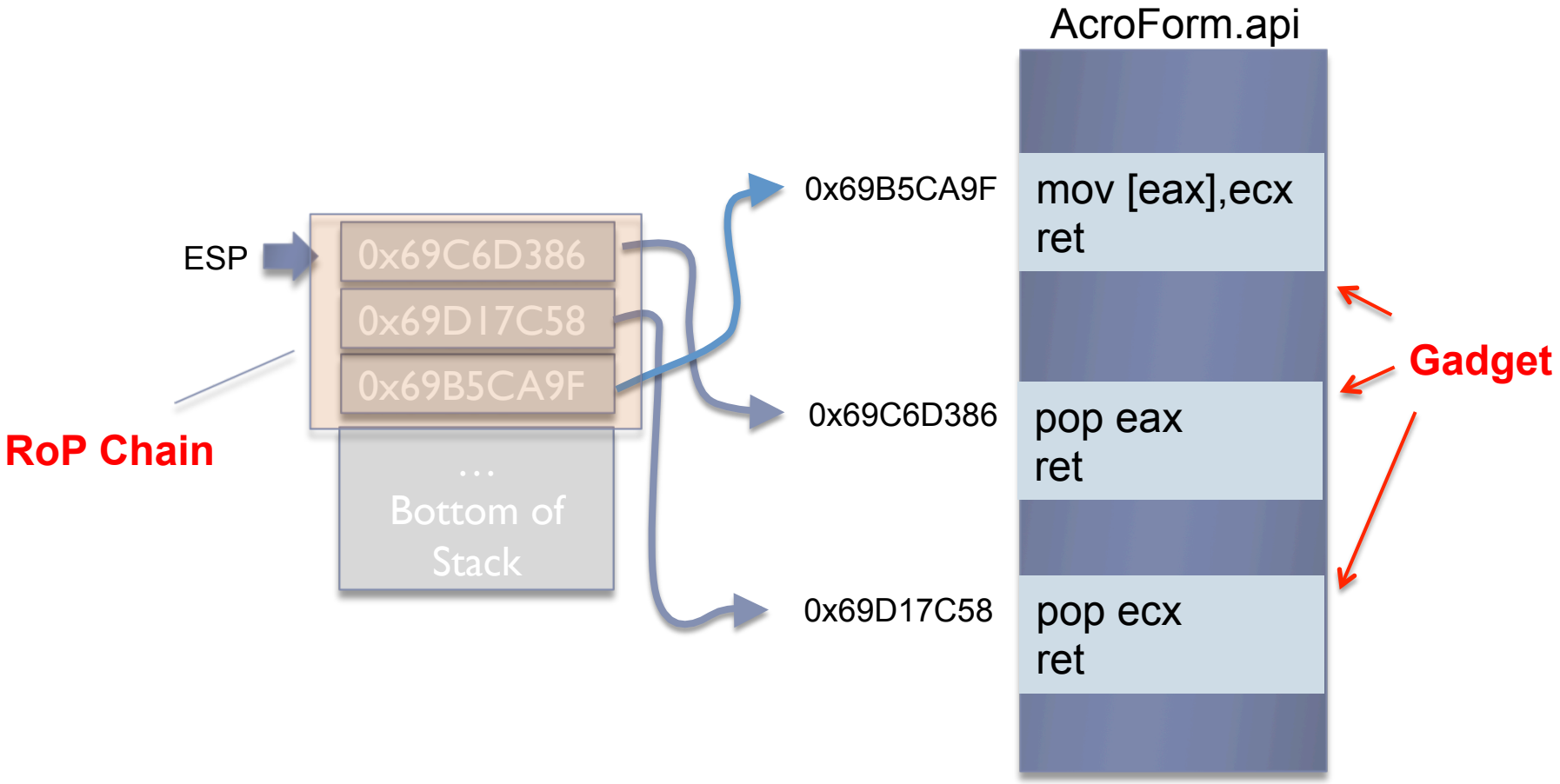
Scan RoP from Memory Dump

- What's in a full context process memory dump
  - Pages from all segments of process memory (stack, heap, code)
  - Shared libraries (DLLs)
  - Dump of registers (especially EIP, ESP)
  - Meta-data
    - Names of the shared library (DLL)
    - Call stack (backtrace)

Scan RoP from Memory Dump (cont.)

- Find a memory region that contains chain of gadget addresses.
- Gadget address has a following "ret" or indirect jmp *nearby* (e.g within 16 instructions after the address).

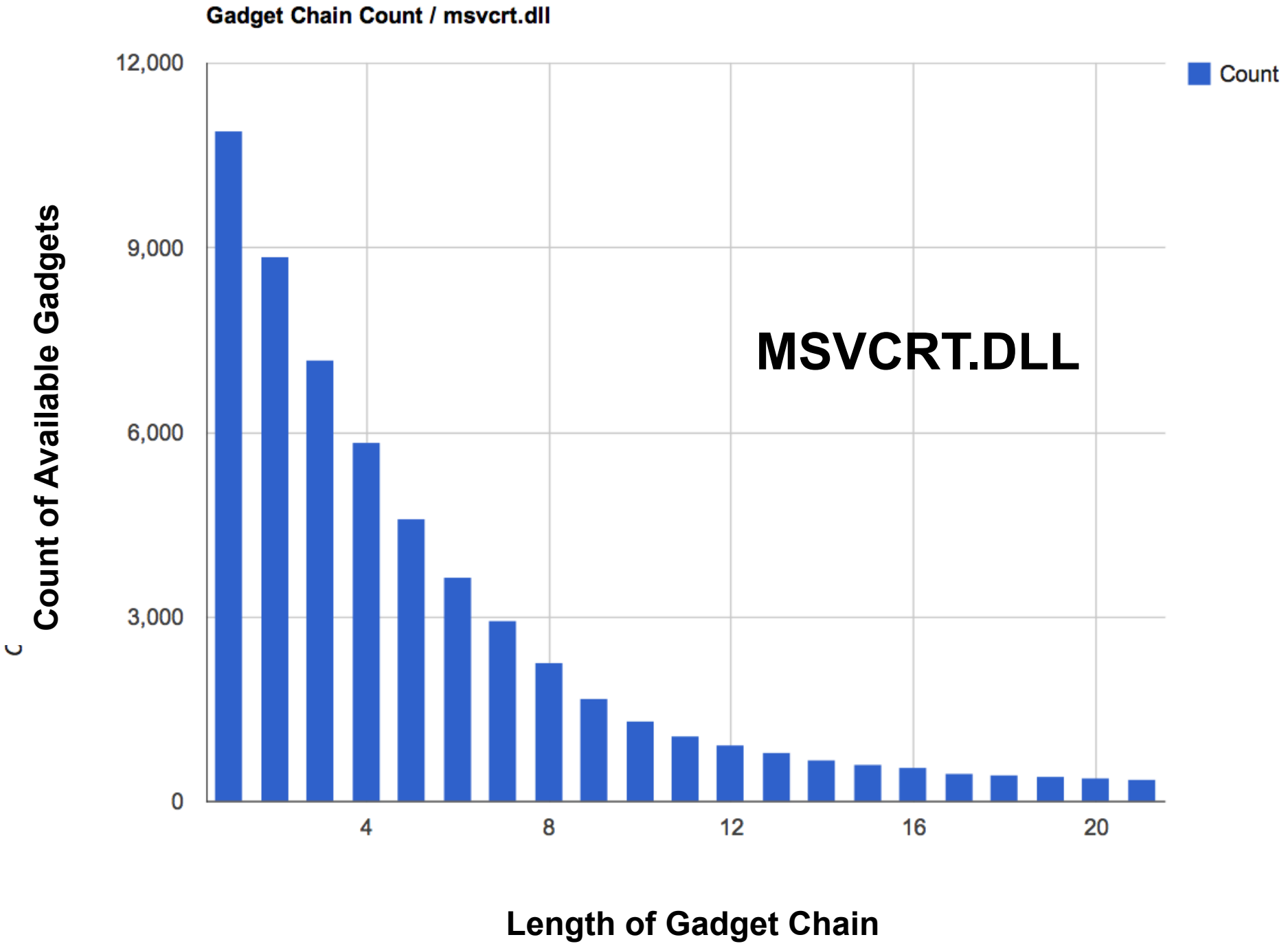
Characteristics of RoP Gadget

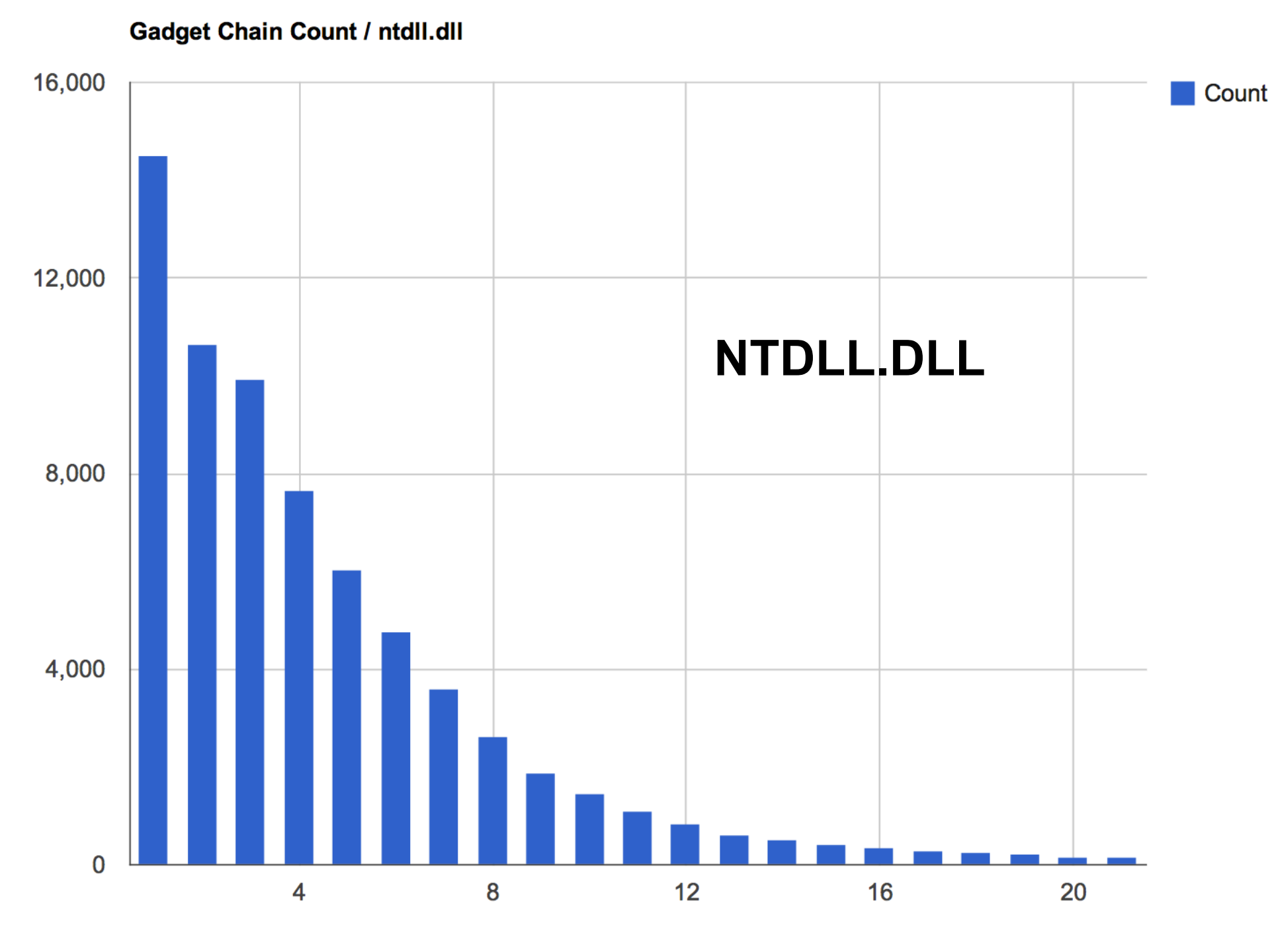


A RoP gadget by definition ends with “ret” (or a jump)

How to Search for RoP Chains

- Check every 4-bytes
  - Searching memory for values that look like an address that points to an executable section.
  - If yes, check if that place has a “ret” somewhere nearby.
  - How far is nearby?


Gadget Chain Count / msvcrt.dll
Count
12,000
9,000
6,000
3,000
0
Count of Available Gadgets
MSVCRT.DLL
4
8
12
16
20
Length of Gadget Chain


Gadget Chain Count / ntdll.dll
16,000
12,000
8,000
4,000
0
4
8
12
16
20
Count
NTDLL.DLL

Challenge #1: Efficiency

- Scan the whole memory dump
  - Slow!
  - An IE8 full context dump ~ 0.5GB

- Core problem is to find where is the "gadget stack"
  - Search only places that can possibly hold gadget addresses.
    - In theory, any data can be used as a gadget chain.
    - In practice, chains are located in writable pages.

  - Other heuristics to speed up
    - if stack pivot occurs, focus scan on new stack

Challenge #2: Accuracy

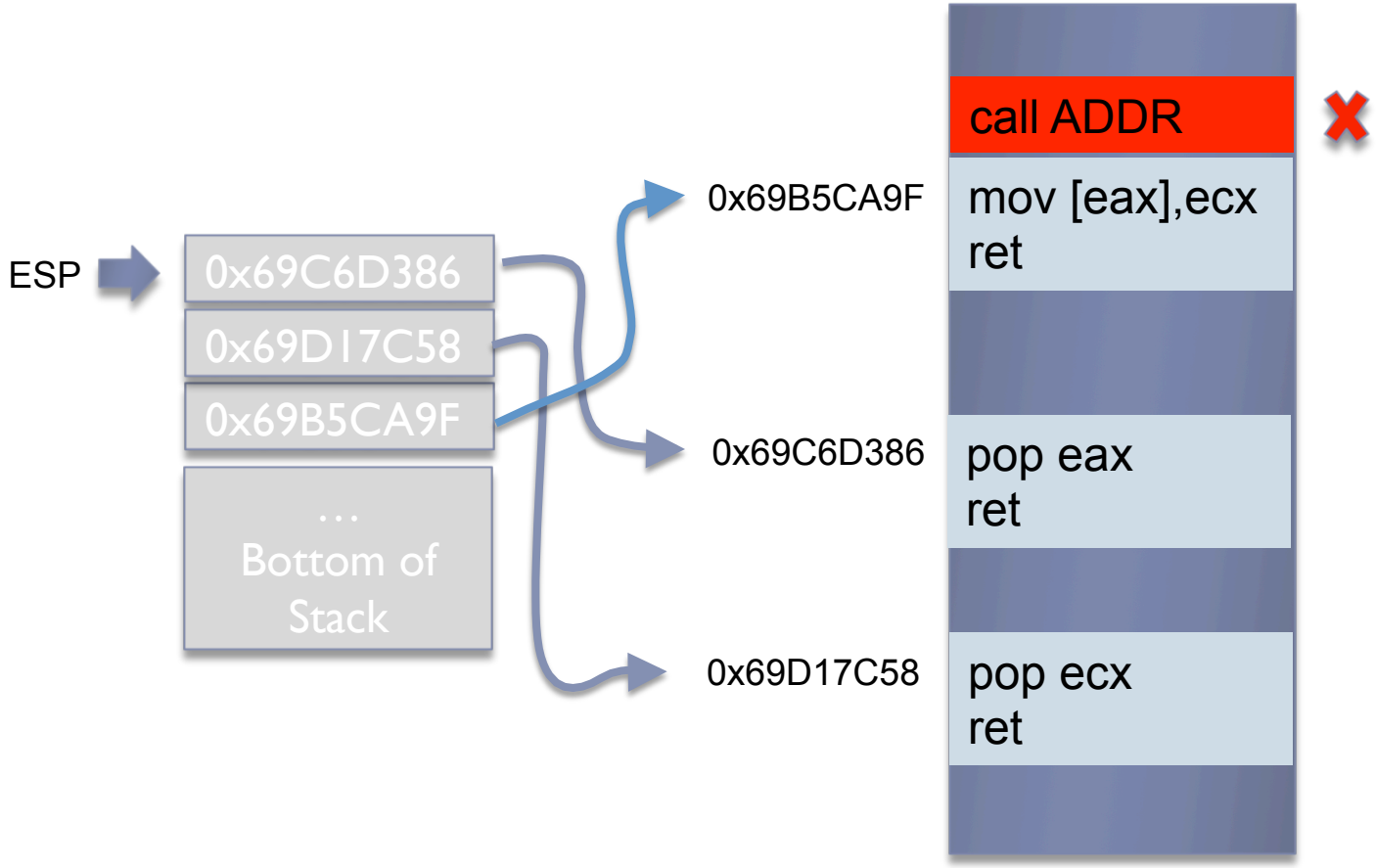
- False Positives
  - Regular calls
    - Return addresses on stack
    - Points to code section and might have a “ret” nearby

Challenge #2: Accuracy

- False Positives
  - Regular calls
    - Return addresses on stack
    - Points to code section and might have a "ret" nearby
  - Solution: Use the heuristics from kBouncer (e.g. call-preceded address)

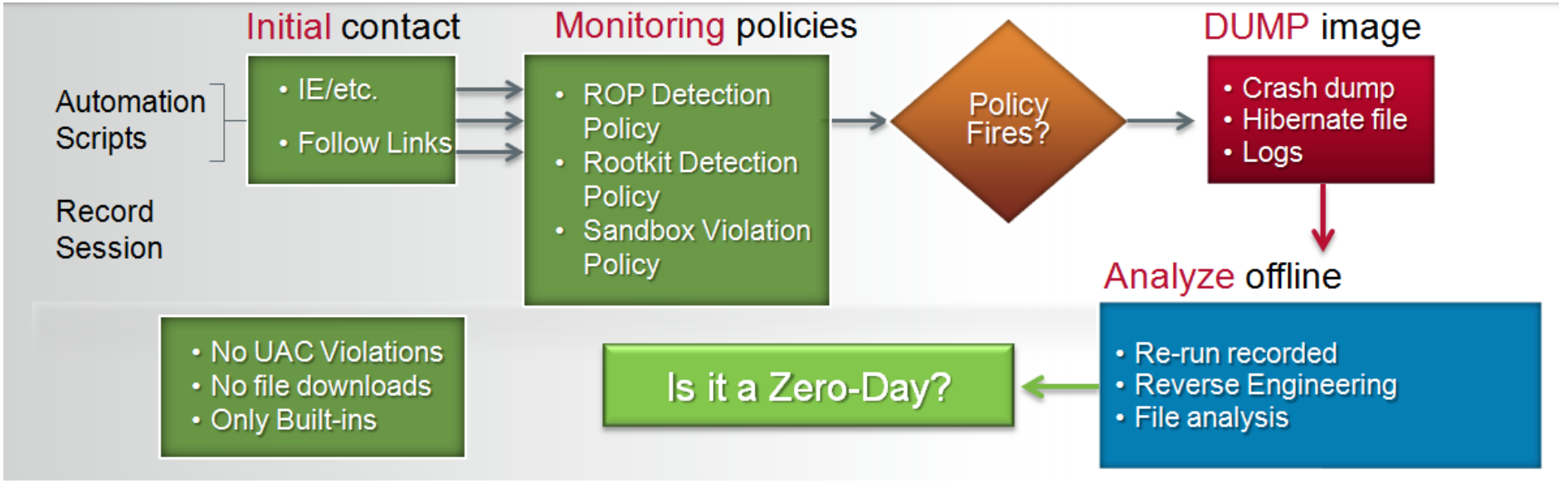
Avoid False Positives

Detect return addr for regular calls



How/when to generate memory dump?

Browser Exploit Detection Flow



With patched OS, exploits are not triggered even ROP chain in memory

Why RoP Discovery is Important

- RoP chain is in memory but never gets executed
  - System being patched
  - Environment changes
  - Attacker choice
- RoP execution was not detected

ROP with Stack Pivoting Detection

- Stack Pivoting needs to point the stack pointer to customized data buffer, usually in heap.

- Current detection solutions
  - Critical APIs check
    - Windows 8
    - ROP guard
    - Different HIPS solutions

Problems in API based Detection

- Known Problems
  - Hook hopping could bypass the check
  - Valid stack pointer before API calling will bypass the check
  - Many ongoing effort …
- Improvement
  - Check during every syscall
  - Check during single step
  - Check during single step + BTF

Stack Pivoting Detection with BTF

- BTF is the flag in MSR_DEBUGCTLA MSR
- Used to enable single-step on branches

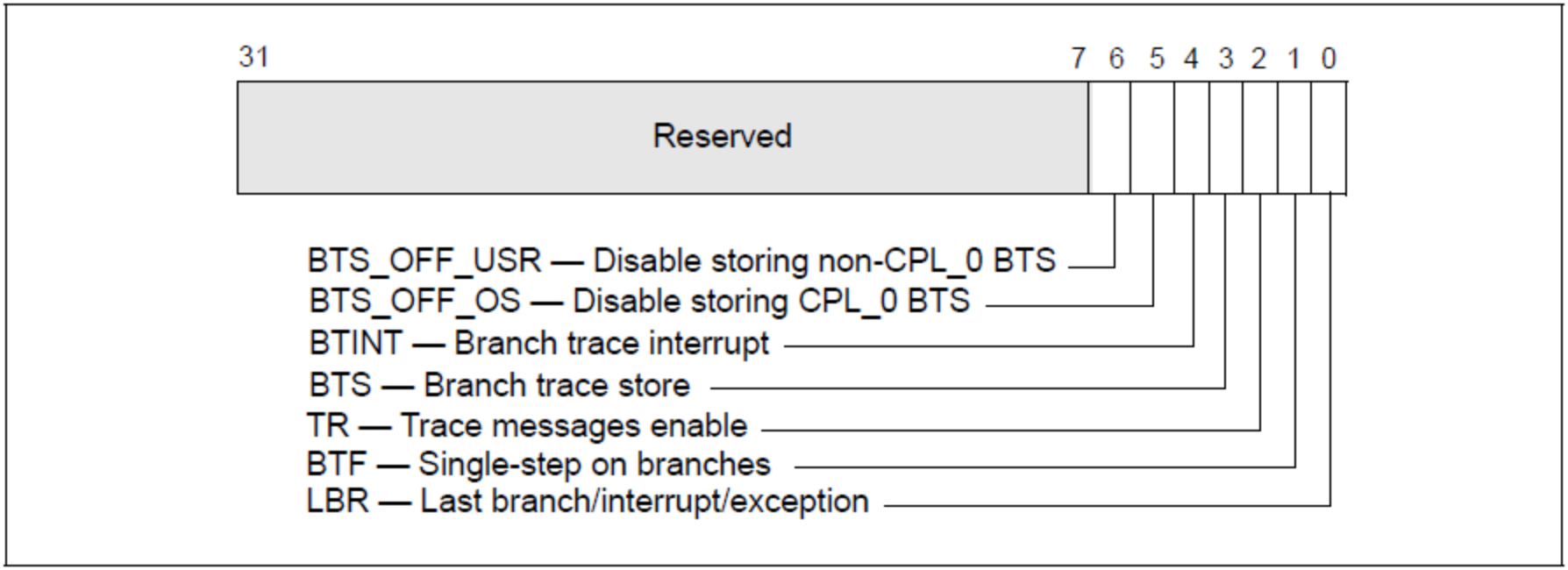
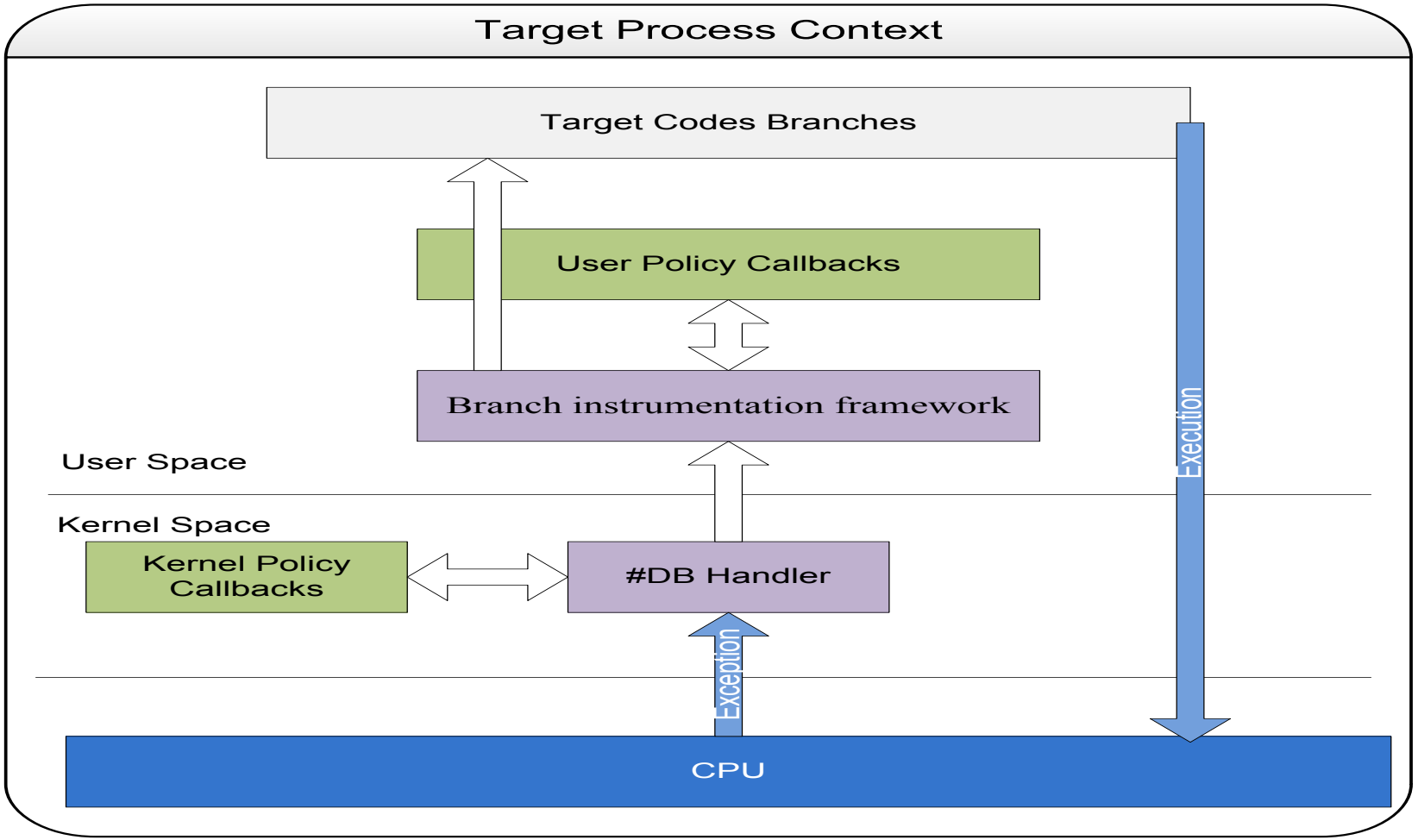


Figure 17-12. MSR_DEBUGCTLA MSR for Pentium 4 and Intel Xeon Processors

BTF Branch Trace Example



Case Study (APSA13-02)

- Reported by FireEye in February 2013
- Best Client-Side Bug for CVE-2013-0641 (Pwnie 2013)
- Sophisticated ROP only without shellcode
- Adobe Sandbox Bypassing: first publicly and wildly used exploitation

Demo Time

Ongoing Work

- UI improvement
  - Better representation of RoP chain candidates
- Traceback to vulnerable code
  - e.g.
    - LBR log
    - Call stacks prior exploit
- Decompiler
  - RoP chain to code

Summary

- RoP chain discovery is desirable in malware analysis.
  - Detecting the first stage of modern day exploitations.
- unRoP searches for RoP chain from memory dumps.
  - Relies on gadget characteristics (short, and no call preceded)
  - Will need to improve as RoP attacks evolve

Thanks!

kangli@uga.edu
ldpatchguard@gmail.com
lee2704@uga.edu