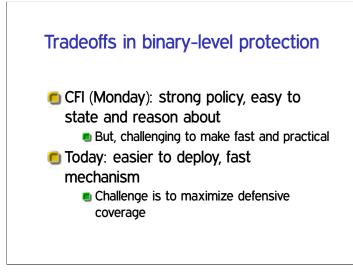
#### 8271 discussion of: "Transparent ROP Exploit Mitigation Using Indirect Branch Tracing"

Stephen McCamant (Original paper: Vasilis Pappas, Michalis Polychronakis, and Angelos D. Keromytis) University of Minnesota (Original paper: Columbia University)

#### Outline

#### Background

- LBR-based approach
- Administrative break
- kBouncer implementation
- Limitations and counterattacks



# Microsoft BlueHat contest

- Contest for new ideas in memory-safety defenses
- Supply Windows prototype with license to Microsoft
- No more than 5% overhead, no "application compatibility regressions"
- 🖲 Due date was April 1st 2012

# BlueHat results

- 1st This paper's project, won \$200k
- 2nd ROPGuard [34], \$50k, used in next version of EMET
- 3rd \$10k + MSDN subscription, also anti-ROP

# Review: ROP

- Create attacks by reusing small pieces of existing code
- Connected by returns or other indirect jumps
- Evolved from return-to-libc, Shacham coined name and demonstrated Turing completeness

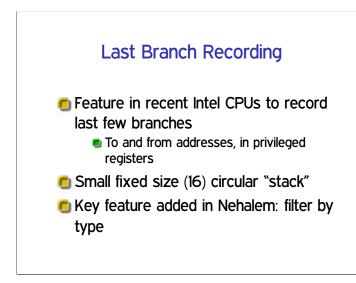
## ROP in the arms race

- W⊕X (e.g. DEP) and ASLR widespread but incomplete
- Most attacks use ROP to circumvent W⊕X
- Defensive next step: do something about ROP

#### Outline

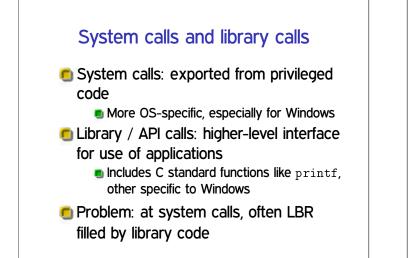
#### Background

- LBR-based approach
- Administrative break
- kBouncer implementation
- Limitations and counterattacks



## Sensitive operations

- Too expensive to check for attack constantly
- Intuition: attacks involve effect outside subverted process
- So, attack must add or change a system call
- Some particularly useful for attacks



#### In-process hooking

- Add extra code to run at start of library function
- Offensive and defensive technique
- Typical approach: overwrite first few code bytes
- Problem: in-process security checks can be bypassed

# Checkpointing approach

- Use hook to check for ROP on library entry point
- If no ROP detected, store "checkpoint" record
- On system call, verify appropriate checkpoint, then clear

## Detection: returns not to call sites

- Calls and returns don't always match correctly
  - **E.g.** longjmp, **user-space threads**, etc.
- But, the target of a return is an instruction directly after a call
- Approach: check if each return address is preceded by a call
- **(**) Any return without call  $\rightarrow$  detect attack

# Kinds of gadgets

- Most convenient gadgets are short and end in return
- Gadgets ending in non-return jumps (JOP) demonstrated in theory
   But not common in current attacks
- Long gadgets harder to program with
- This paper's definition: up to 20 instructions, need not be contiguous

## Detection: chaining of gadgets

- Hypothesis: ROP has longer chains of shorter code segments than benign code
- Detect attack if at least 8 consecutive LBR entries are all gadget-sized
- Maximum observed in benign code: 5 before sensitive calls, 9 anywhere

#### Outline

#### Background

LBR-based approach

Administrative break

kBouncer implementation

Limitations and counterattacks

#### Upcoming topics

- 2/17 Smartphone security
- 2/24 Web application security
  - 3/3 (Anti-)censorship

3/10 Tor

After spring break: rough ideas, will finalize after finding more papers

# **Project meetings**

- Purpose: discuss project topics
- 🖲 Email me to set up
- This Friday is the last day

#### Outline

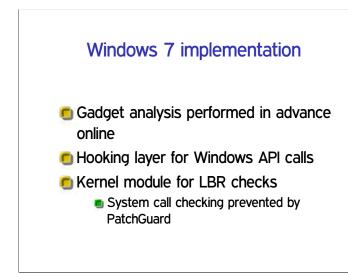
Background

LBR-based approach

Administrative break

kBouncer implementation

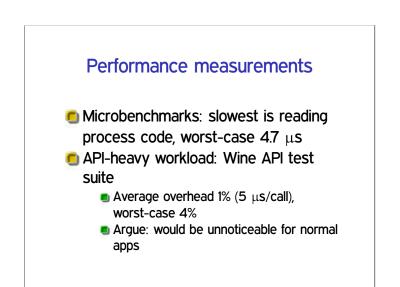
Limitations and counterattacks



## **Pin-based simulator**

- Simulate LBR in software using dynamic translation

   Implemented with Pin, a commonly-used framework
- Collect statistics on benign software, used in design
- Not fast enough to use as a practical defense





## Outline

Background

LBR-based approach

Administrative break

kBouncer implementation

Limitations and counterattacks

#### **API selection**

- Paper gives 52 currently protected functions
- Empirically, a blacklist this long is usually found later to be missing entries
- Protect all API calls? Paper argues would be excessive

#### LBR size

 LBR size is fixed in hardware
 Might get bigger in subsequent chips
 However, even in LBR size were unlimited, might still want to limit

checks for performance Problem: attackers don't care about

performance; if limit is k, adversary uses k+1

## Is ROP still possible?

- Thorough analysis: "part of future work"
- 6.4% of gadgets are call-preceded, (3% of shorter 5-byte ones)
- One automated system fails when limited to 20% of gadgets

But human attackers can be more creative

# Gadget size

- To stop chaining detection, find a few large gadgets
- Increasing detection length beyond 20 might bring false positives
- Think about reusing larger chunks of code, not just gadgets
  - E.g., subvert code that calls a sensitive function

#### Attacks against host IDSes

- C.f. "Automating Mimicry Attacks Using Static Binary Analysis", USENIX'05
- Attacker has taken over binary, but must conceal attack system calls
- Key challenge: how to regain control after system call with legitimate-looking call stack
- Short answer: overwrite indirect jump pointers