Can Data-Only Exploits be Detected at Runtime Using Hardware Events?

A Case Study of the Heartbleed Vulnerability

Gildo Torres & Chen Liu

Department of Electrical and Computer Engineering

Clarkson University

Hardware and Architectural Support for Security and Privacy (HASP)

June 18th, 2016

Clarkson UNIVERSITY defy convention

Wallace H. Coulter School of Engineering

Outline

- Background
  - Control Exploits vs Data Exploits
  - Hardware Performance Counters
- Motivation
- Heartbleed Vulnerability
- System Architecture
- Experiments
  - Hardware Event distributions
  - Detection Accuracy
- Conclusion

Background

Control Exploits:

- Exploit vulnerabilities using a **payload** to **execute arbitrary code**
- **Hijack control-flow** of the victim program

Data Exploits:

- **Conserve control-flow** of victim application
- Achieve **same level of compromise** of target systems

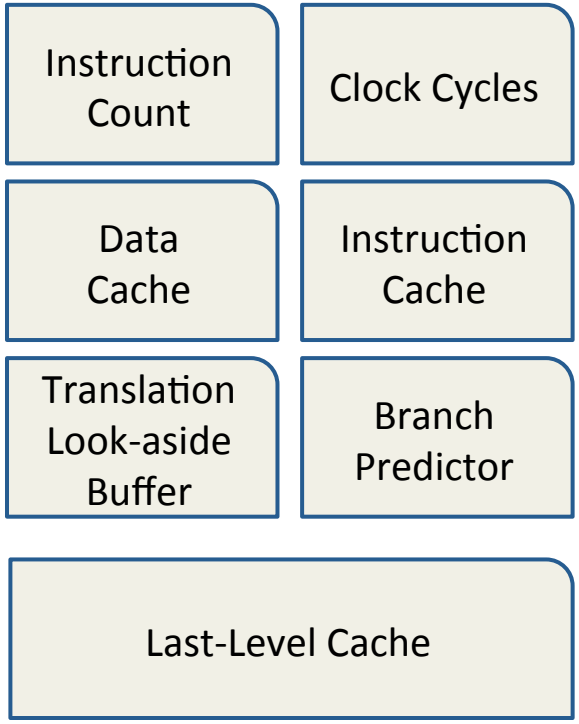
Hardware Performance Counters (HPCs)

What are these:

- Special HW registers available on most modern processors
- Over 200 measurable HW conditions

Benefits:

- Very **fast** to access
- **Difficult** for attackers to **manipulate**
- Capture **raw execution** behavior



Motivation

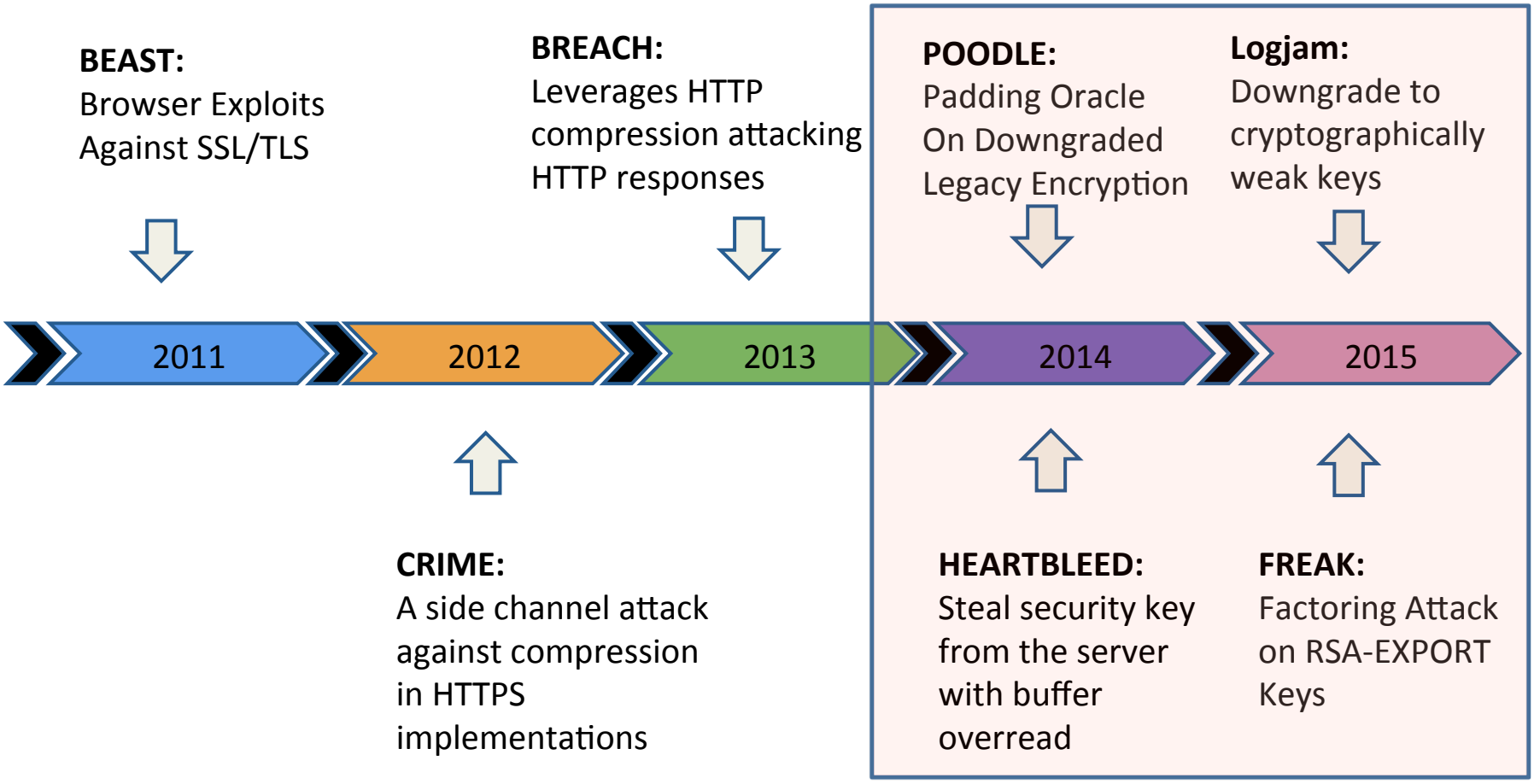
Previous research:

- Signature-based detection
- Rootkit detection using HPCs to monitor syscalls (Wang DAC'13)
- HPCs for detection of malware (Demme ISCA'13, Tang RAID'14)

This work:

How effective is hardware level information for the detection of **Data Exploits**?

Attacks Against TLS/SSL



Heartbleed Vulnerability

What is it?

OpenSSL vulnerability within heartbeat Extension for the TLS/DTLS protocols

The problem:

Missing check between an advertised request size and the real token size

Implications:

Allows malicious party to trick the target into sending more information (memory content) than it should

How does it work?

Mismatch between the real size of a message's token (Tsize) and the size of the payload that is advertised (Rsize).

Heartbeat request

...	Type (1 byte)	Length (2 bytes)	Data (variable bytes)
		Rsize	Tsize

Malicious request

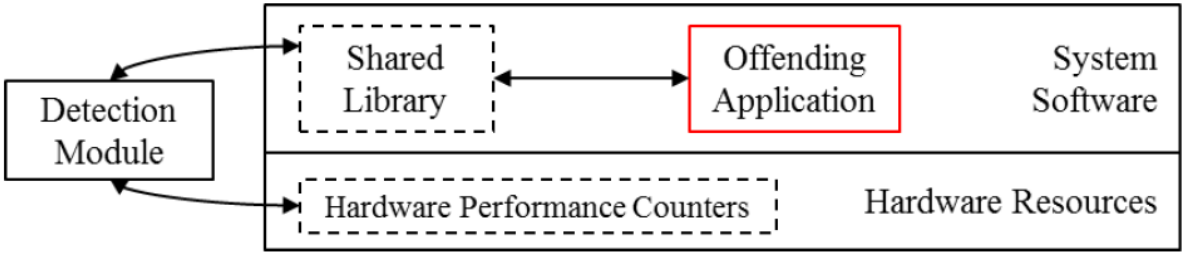
...	HB_REQUEST	65535	1 random byte

Leaked data = Rsize – Tsize => 64 KB

System Architecture

Goal:

Investigate feasibility of using an **anomaly-based detection** scheme that utilizes information collected from hardware performance **counters** at **runtime** to detect data-oriented attacks in user space libraries



Experimental Setup

Platform:

- Intel Core i7-950 (Nehalem, Quad-Core, HT, 3.06GHz)
- Linux kernel version 3.8.0

Vulnerability:

- OpenSSL version 1.0.1f (Heartbleed)

Tools:

– Linux Perf_events interface (syscall)

Hardware Events monitored:

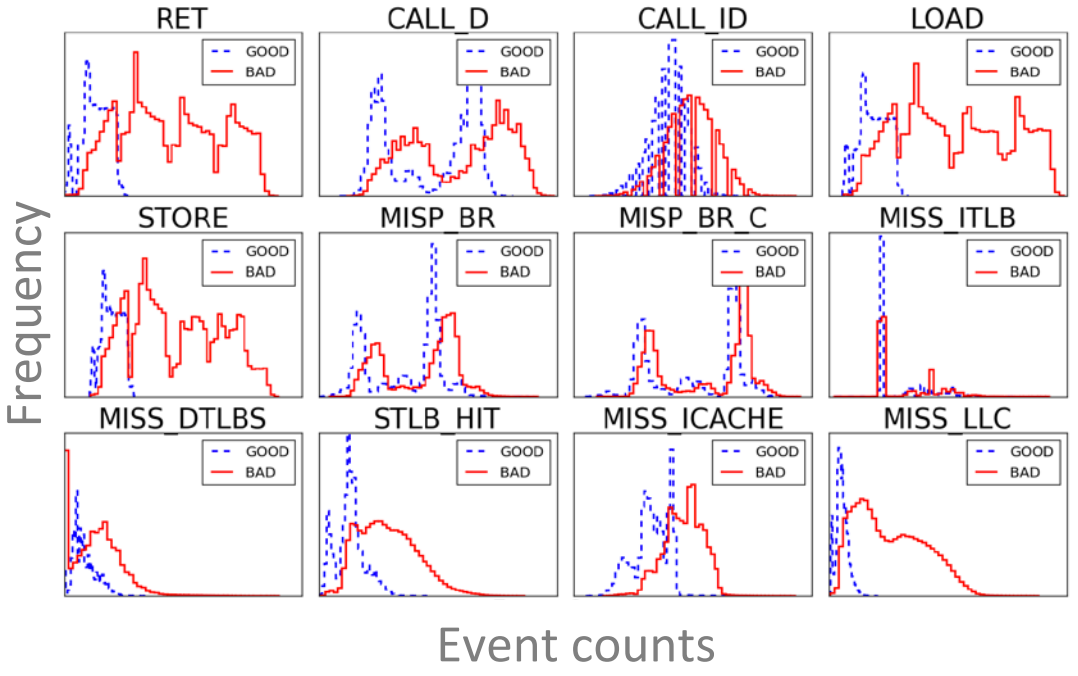
Event Name	Description
RET	Near return instructions retired
MISP_BR	Mispredicted branch instructions
LOAD	Load instructions retired
MISP_BR_C	Mispredicted conditional branches
STORE	Store instructions retired
MISS_ITLB	I-TLB misses
STLB_HIT	Shared TLB hits after i-TLB misses
MISS_DTLB	DTLB-misses
CALL_ID	Indirect near call instructions retired
MISS_ICACHE	I-Cache misses
CALL_D	Direct near call instructions retired
MISS_LLC	Last Level Cache misses

Malicious vs Legitimate Distribution

Different degrees of overlapping:

Some events noticeably different:
RET, LOAD, STORE, MISS_LLC

Some events barely distinguishable
MISP_BR, MISP_BR_C, MISS_ITLB

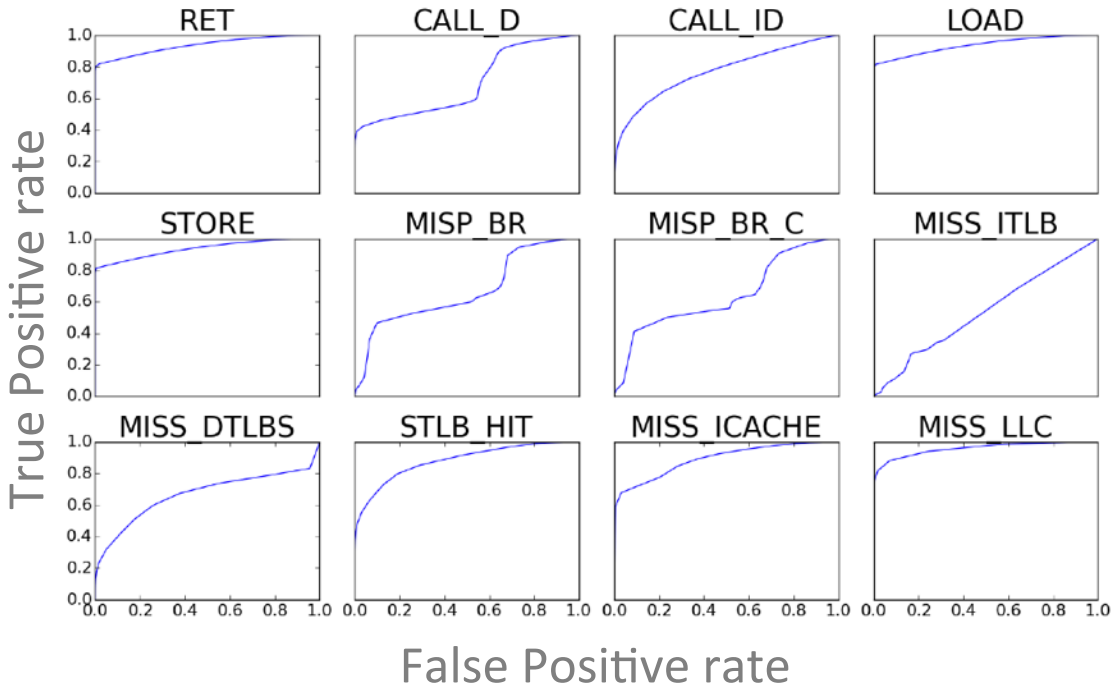


Detection Accuracy (1)

Receiver Operating Curves (ROC)

True Positive to False Positive ratio of different classification thresholds

Individual performance represented by Area Under the Curve (AUC)

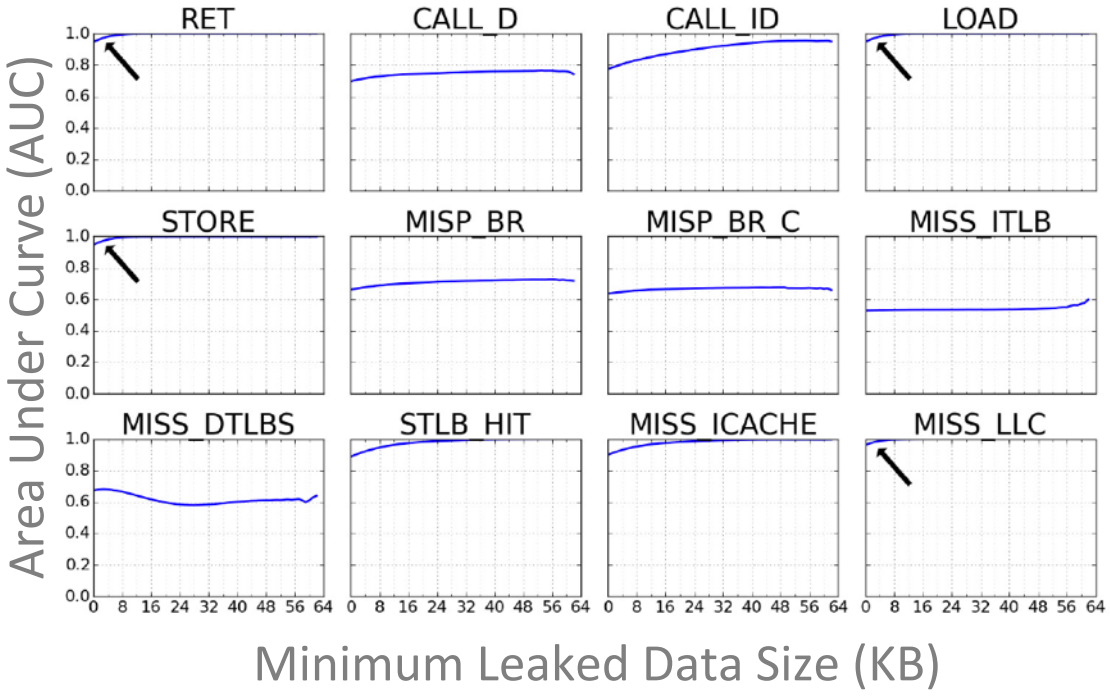


Less overlapping of distribution ➔ better classification performance

Detection Accuracy (2)

Area Under Curve

Extended study of classification to **leak gap** ranging between 1KB - 64KB



Higher detection accuracy as the **gap size grows:** *RET, LOAD, STORE, MISS_LLC*

Some events **immune** to growing **gap size:** *CALL_D, MISP_BR, MISP_BR_C, MISS_ITLB, MISS_DTLBS*

Detection Accuracy (3)

Support Vector Machine (SVM)

- Two-class SVM: Training set containing both ***good*** and ***bad*** requests
- One-class SVM: Training set exclusively containing ***good*** requests

Classifier	Classification Accuracy (%) of different sets						
	≥ 1 byte	≥ 1KB	≥ 2KB	≥ 4KB	≥ 8KB	≥ 16KB	≥ 32KB
2-class SVM	92.8	94.02	95.38	97.01	98.75	99.98	100
1-class SVM	70.88	73.04	73.7	74.68	74.55	74.46	74.41

0.99% False Negative rate

Detection Accuracy (4)

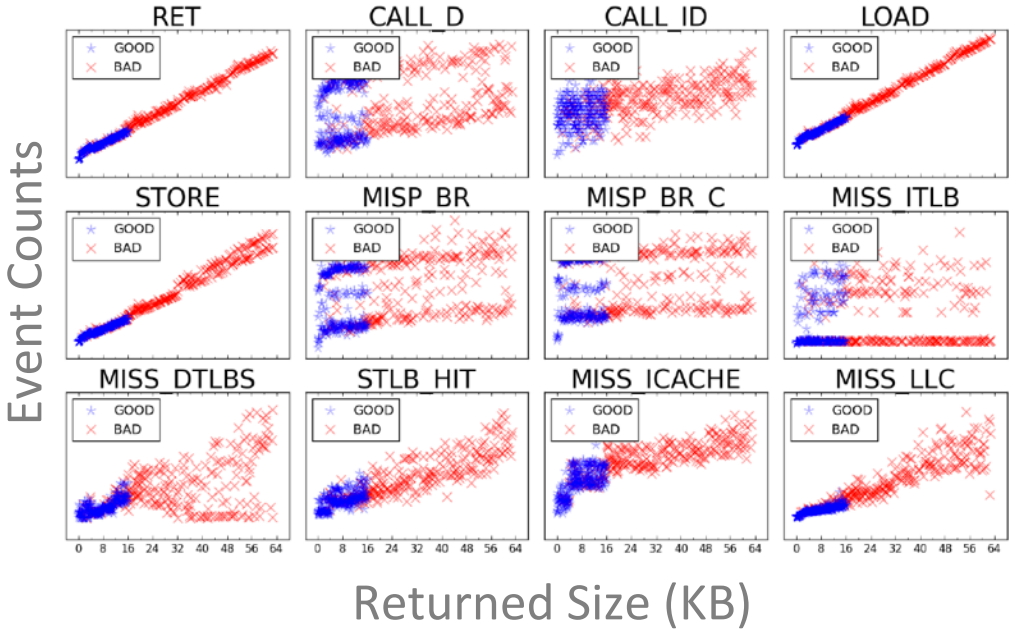
Hardware Event Subsets

- Studied individual 6 most effective HW events:

  [RET, LOAD, STORE, MISS_ICACHE, STLB_HIT, MISS_LLC]

- Classification rates **improved** as the gap **size grows** larger
- Classification average:
  - **Worst:** 96.8% *[LOAD, STORE, STLB_HIT, MISS_LLC]*
  - **Best:** 97.8% *[RET, LOAD, STLB_HIT, MISS_ICACHE]*

Hardware Events Behavior Analysis



Behavioral Instructions	Instructions Retired - Loads, Stores - Indirect Calls, Returns - Direct Calls - Branches Taken^{+} — Conditional Branches Taken^{+}
Behavioral Data	Memory Operands - Reads^{+}, Writes^{+}
Models	Mispredicted Branches - Mispredicted Conditional Branches Last Level Cache Misses - I-Cache Misses, D-Cache Misses^{+} TLB Misses - I-TLB Misses, D-TLB Misses - Shared TLB Hits after I-TLB Miss

Conclusions

- Experiments suggest that **Data Exploits** are **harder to detect** using low-level **hardware events**
- Study showed that **different events** experienced **different sensitivity** to the studied attack
- Non-deterministic events showed potential for differentiating between normal and abnormal behavior

THANK YOU!

QUESTIONS?