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
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

BEAST:
Browser Exploits Against SSL/TLS

BREACH:
Leverages HTTP compression attacking HTTP responses

POODLE:
Padding Oracle On Downgraded Legacy Encryption

Logjam:
Downgrade to cryptographically weak keys

2011

CRIME:
A side channel attack against compression in HTTPS implementations

2012

HEARTBLEED:
Steal security key from the server with buffer overread

2013

FREAK:
Factoring Attack on RSA-EXPORT Keys

2014

2015
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

<table>
<thead>
<tr>
<th>...</th>
<th>Type (1 byte)</th>
<th>Length (2 bytes)</th>
<th>Data (variable bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rsize</td>
<td>Tsize</td>
<td></td>
</tr>
</tbody>
</table>

Malicious request

<table>
<thead>
<tr>
<th>...</th>
<th>Type</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HB_REQUEST</td>
<td>65535</td>
<td>1 random byte</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RET</td>
<td>Near return instructions retired</td>
</tr>
<tr>
<td>MISP_BR</td>
<td>Mispredicted branch instructions</td>
</tr>
<tr>
<td>LOAD</td>
<td>Load instructions retired</td>
</tr>
<tr>
<td>MISP_BR_C</td>
<td>Mispredicted conditional branches</td>
</tr>
<tr>
<td>STORE</td>
<td>Store instructions retired</td>
</tr>
<tr>
<td>MISS_ITLB</td>
<td>I-TLB misses</td>
</tr>
<tr>
<td>STLBI_HIT</td>
<td>Shared TLB hits after i-TLB misses</td>
</tr>
<tr>
<td>MISS_DTLB</td>
<td>DTLB-misses</td>
</tr>
<tr>
<td>CALL_ID</td>
<td>Indirect near call instructions retired</td>
</tr>
<tr>
<td>MISS_ICACHE</td>
<td>I-Cache misses</td>
</tr>
<tr>
<td>CALL_D</td>
<td>Direct near call instructions retired</td>
</tr>
<tr>
<td>MISS LLC</td>
<td>Last Level Cache misses</td>
</tr>
</tbody>
</table>
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 \textit{good} and \textit{bad} requests
• One-class SVM: Training set exclusively containing \textit{good} requests

\begin{table}
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Classifier} & \multicolumn{6}{|c|}{\textbf{Classification Accuracy (\%) of different sets}} \\
\hline
 & \text{\$\geq 1\ \text{byte}\$} & \text{\$\geq 1\text{KB}\$} & \text{\$\geq 2\text{KB}\$} & \text{\$\geq 4\text{KB}\$} & \text{\$\geq 8\text{KB}\$} & \text{\$\geq 16\text{KB}\$} & \text{\$\geq 32\text{KB}\$} \\
\hline
2-class SVM & 92.8 & 94.02 & 95.38 & 97.01 & 98.75 & 99.98 & 100 \\
\hline
1-class SVM & 70.88 & 73.04 & 73.7 & 74.68 & 74.55 & 74.46 & 74.41 \\
\hline
\end{tabular}
\end{table}

0.99\% False Negative rate
Detection Accuracy (4)

Hardware Event Subsets

• Studied individual 6 most effective HW events:

  ![image]

• Classification rates improved as the gap size grows larger

• Classification average:

  – **Worst:** 96.8%  
  
  ![image]

  – **Best:** 97.8%  
  
  ![image]
Hardware Events Behavior Analysis

Returned Size (KB)

Event Counts

Returned Size

<table>
<thead>
<tr>
<th>Behavioral Instructions</th>
<th>Behavioral Data</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructions Retired</td>
<td>Memory Operands</td>
<td>Mispredicted Branches</td>
</tr>
<tr>
<td>- Loads, Stores</td>
<td>- Reads+, Writes+</td>
<td>- Mispredicted Conditional Branches</td>
</tr>
<tr>
<td>- Indirect Calls, Returns</td>
<td></td>
<td>- Last Level Cache Misses</td>
</tr>
<tr>
<td>- Direct Calls</td>
<td></td>
<td>- I-Cache Misses, D-Cache Misses+</td>
</tr>
<tr>
<td>- Branches Taken</td>
<td></td>
<td>- TLB Misses</td>
</tr>
<tr>
<td>— Conditional Branches Taken+</td>
<td></td>
<td>- I-TLB Misses, D-TLB Misses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Shared TLB Hits after I-TLB Miss</td>
</tr>
</tbody>
</table>
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?