THE FOLLOWING IS INTENDED TO OUTLINE OUR GENERAL PRODUCT DIRECTION. IT IS INTENDED FOR INFORMATION PURPOSES ONLY, AND MAY NOT BE INCORPORATED INTO ANY CONTRACT. IT IS NOT A COMMITMENT TO DELIVER ANY MATERIAL, CODE, OR FUNCTIONALITY, AND SHOULD NOT BE RELIED UPON IN MAKING PURCHASING DECISIONS. THE DEVELOPMENT, RELEASE, AND TIMING OF ANY FEATURES OR FUNCTIONALITY DESCRIBED FOR ORACLE'S PRODUCTS REMAINS AT THE SOLE DISCRETION OF ORACLE.
Application Data Integrity

Dynamic Tainting in Hardware
SPARC @ Oracle
7 Processors in 6 Years

Including Software in Silicon

- App Data Integrity
- DB Query Acceleration
- Inline Decompression
- More....

SPARC T3
16 x 2nd Gen Cores
4MB L3 Cache
1.65 GHz

SPARC T4
8 x 3rd Gen Cores
4MB L3 Cache
3.0 GHz

SPARC T5
16 x 3rd Gen Cores
8MB L3 Cache
3.6 GHz

SPARC M5
6 x 3rd Gen Cores
48MB L3 Cache
3.6 GHz

SPARC M6
12 x 3rd Gen Cores
48MB L3 Cache
3.6 GHz

SPARC M7
32 x 4th Gen Cores
64MB L3 Cache
4.1 GHz

Low Cost SPARC
8 x 4th Gen Cores
16MB L3 cache
IB
Security Everywhere

• Every application is vulnerable
• Exploitable in future
• Security vulnerabilities in unexpected places
• Enable security by default
  – No performance penalty
  – Prevent whole classes of problems
  – No modification to the program
Address space layout

- **Binary**
  - (Text)
  - Data Segment

- **Stack**
- Library 3 (LIBM)
- Library 2 (LIBZ)
- Library 1 (LIBC)
Memory Corruption

Heartbleed - Heap Buffer Overflow
Heartbleed - Impacted Websites Using OpenSSL

Heartbeat request sent to victim

<table>
<thead>
<tr>
<th>Type</th>
<th>Payload size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>HB_REQUEST</td>
<td>65535</td>
<td>Hello</td>
</tr>
</tbody>
</table>

Victim responds with requested payload size (64K bytes)

<table>
<thead>
<tr>
<th>Type</th>
<th>Payload size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>HB_RESONSE</td>
<td>65535</td>
<td>Hello</td>
</tr>
</tbody>
</table>

Payload_size does not match Payload

Unauthorized data returned to requestor
Application Data Integrity
(ADI or ADP or MCD or SSM)
Security In Silicon – Silicon Secured Memory
Always-On Memory Protection in Hardware

- Story begins in 2010 with Oracle acquiring Sun Microsystems
- First ever hardware-based memory intrusion protection of its kind
- Always-On hardware approach has near zero performance impact
- Stops programs from accessing other applications memory
  - Stops Malicious Programs like Venom and Heartbleed
  - Helps Developers Find Difficult Bugs
M7 SSM Would Have Detected Heartbleed in Real-Time

• When the hacker request attempted to read data beyond its allotted buffer limits (buffer over-read), ADI would have stopped the access and generated a signal.
• The signal handler could have responded by flushing the data and terminating the offending connection.
**Security In Silicon: Silicon Secured Memory**

- **Protects data in memory**
- Hidden “color” bits added to *pointers* (key), and content (lock)
- Pointer color (key) must match content color or program is aborted
  - Set on *memory allocation*, changed on *memory free*
  - Protects against *access off end of structure, stale pointer access* and malicious attacks
Silicon Secured Memory: Buffer Overflows

Any Processor

Memory

Applications

Pointer

SPARC M7 Processor

Memory

Applications

STOP

Pointer

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.
Demo of ADIHEAP

- Demo of OpenSSL Heartbleed security vulnerability
- 64-bit processes only
- T7/M7/S7 or higher
- For production use: libc or libadimalloc
  - sxadm exec -s adiheap=enable
  - link with -z sx=adiheap=enable
  - elfedit -e 'dyn:sunw sx ADIHEAP enable'
- For development use: libdiscoverADI from Developer Studio
  - instrument with discover -i adi
  - preload via LD_PRELOAD_64
Memory Protection

Related Work
Techniques

• Segmentation (x86)
• Paged Virtual Memory
• Intel Memory Protection Extensions (MPX)
• Memory Protection Keys (MPK)
• Dynamic Tainting (ADI)
ADI version check

- ADI version stored in DIMM modules
- 4 bits “taken” for ADI version from spare ECC bits
- ADI version stored in L1/L2/L3 caches
- ADI version stored in pointers
- MMU performs “version match” check
- precise or disrupting trap
ECC Structure

- 18 DRAM chips in x4 DIMM
- 1 DRAM chip has 4 pins
- 1 memory access = 8 strobes = 32 bits
- 18 DRAM chips per 4 pins = 72 pins
- 64 bytes data + 8 bytes ECC checkbits
- ECC based on Galois Field GF(2^{24})
- 32 checkbits for row parity
- 24 checkbits for inner parity
- 8 bits previously used for pin sparing
- 4 taken for ADI
- DIMM sparing used nowadays

![ECC Diagram](image)
ADI instructions in userspace

• Oracle SPARC Architecture 2015 (MCD)
• Special instructions (and libc functions) to manipulate ADI from a program
  – adi_get_version()
    • ldx
  – adi_set_version()
    • stx
  – adi_memset()
    • stx
  – adi_get_precise()
    • rd
  – adi_set_precise()
    • wr
  – adi_version_nbits()
ADI Aware Allocator

• ADI needs support from the memory allocator to return “versioned” pointers

• Enable ADI based on:
  – System global setting (sxadm)
  – Per program setting (sxadm or SUNW_SX_ADI* tag)
  – Preloading special ADI allocator

• 16 possible ADI versions
  – Version “0” means no version at all
  – Version 14 is taken for ECC recovery by HW
  – Version 15 means universal match
  – => 13 usable versions

• Different ADI version allocation strategies
  – Solaris libc
  – Developer Studio (libdiscoverADI)
Enabling ADI

• Allocator tells kernel via \textit{adi\_set\_enabled()} syscall:
  – Enable for (virtual) processor: PSTATE.mcde = 1

• Every page entry in Translation Storage Buffer (TSB) needs to have
  • TTE.mcd = 1

• ADIHEAP: enable for data segment and anonymous pages
• ADISTACK: enable for main stack and threads’ stacks
• ADIKERNEL: enable for kmem caches
• other work in progress
Application Data Integrity

ADISTACK
Demo of ADISTACK

- As usually, 64-bit processes only
- T7/M7/S7 or higher
- Enable with:
  - `sxadm exec -s adistack=enable`
  - `link with -z sx=adistack=enable`
  - `elfedit -e 'dyn:sunw_sx ADISTACK enable'`
- Stack walking, stack unwinding
sparcv9 stack layout

foo:
  %l, %i save area
  local stack variables
  temporaries
  %0. save area

bar:
  %l, %i. save area
  local variables
  temps

addresses:
%10
%11
...
%l7
%10
%15
...
%i6 = %fp
%17 = %ret
Return Oriented Programming (ROP)

- sub $g0, $i0, $i0
- ba 0x3ffffc8
- restore
- ...

- sub $l1, $l3, $l1
- stx $l1, [%i2]
- ret
- ...

- xor $l1, $o1, $l1
- or $g1, $l1, $g1
- ret
- ...

- ta 0x40
- ...

gadgets→
What’s still left?

- Procedure Linkage Table (PLT)
- Global Offset Table (GOT)
- ...

Procedure Linkage Table I.

1. \texttt{printf@plt}
   
2. \texttt{ld.so.1}

3. \texttt{resolve\_symbol}
   \texttt{update PLT}
   \texttt{call printf@libc}

4. \texttt{libc}
   \texttt{printf: return}

\texttt{caller}

\texttt{\ldots call printf@plt \ldots}
Procedure Linkage Table II.

caller

... call printf@plt...

1.

PLT

... printf@plt:
call printf@libc...

2.

libc

printf:
... return
Protecting PLT

- Full RELRO (beware the startup cost)
- State of the art these days
- Partial RELRO with ADI (sparc) + MPK (Intel)
ADI Limits and Caveats

• 64-bit processes only

• Performance impact
  – Negligible for the default disrupting traps
  – Optional precise traps for store mismatches have a noticeable impact, should only be used for debug
  – Updating versions is negligible
  – 64 bytes granularity

• Normalize pointers before
  – compare
  – arithmetical operations

• ADI has a high probability of catching bugs, but a bad pointer may accidentally have a matching version

• DMA read (write to memory) resets ADI version to 0
  – Impacts userspace only if Direct I/O is used (UFS userspace, Infiniband)
Q&A Session