# LO-PHI

# Low-Observable Physical Host Instrumentation for Malware Analysis


Chad Spensky[*†], Hongyi Hu[*§] and Kevin Leach[*‡]

*cspensky@cs.ucsb.edu* *hongyihu@alum.mit.edu* *kjl2y@virginia.edu*

***lophi@mit.edu***


*The Network and Distributed System Security Symposium 2016*


[*]MIT Lincoln Laboratory [†]University of California, Santa Barbara [§]Dropbox [‡]University of Virginia



This work was sponsored by the Assistance Secretary of Defense for Research and Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Government.
DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

# Outline

- **Overview of LO-PHI**
- **Instrumentation**
- **Semantic Gap Reconstruction**
- **Automated Binary Analysis**
- **Evaluation (Windows Malware)**
- **Summary**
- **Demo** (Time Permitting)

# The Problem

- **Binary dynamic analysis is becoming increasingly difficult in security-critical scenarios**
  - **Environment-aware malware** can detect various ***artifacts*** exposed by most existing dynamic analysis frameworks and leverage them to avoid detection, or subvert the analysis all together
  - **The observer effect**, i.e. the effects of the measurement itself, can interfere with the analysis, making the results untrustworthy
    - E.g., software-based instrumentation may result in a different memory layout

# The Problem

- ***Introspection*** techniques offer solutions that have fewer artifacts, but must also bridge the ***semantic gap***
  - i.e., translate low-level data to semantically rich output for analysis

# Introspection Options

- Software
  - Pros: cheap, easy to implement
  - Cons: OS dependent, can affect analysis, easily subverted

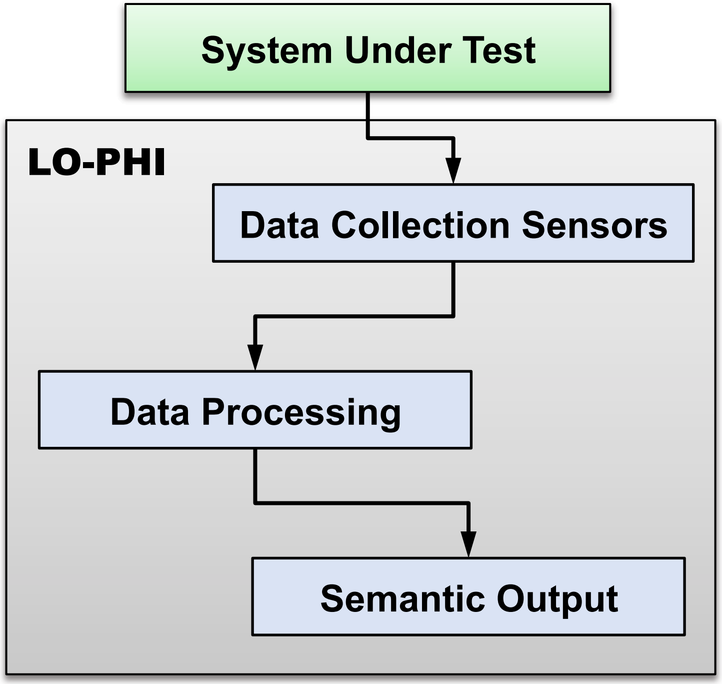


- Virtual machines
  - Pros: development in software, scalable
  - Cons: easily detectable artifacts (E.g. *Redpill*)

- Hardware
  - Pros: potentially very few artifacts, better ground truth
  - Cons: difficult to implement, expensive

# Goals

- **Primary goal**
  - ***Low-Observable Physical Host Instrumentation (LO-PHI)*** aims to obtain ground truth information about a system under test (SUT) while introducing as few artifacts as possible

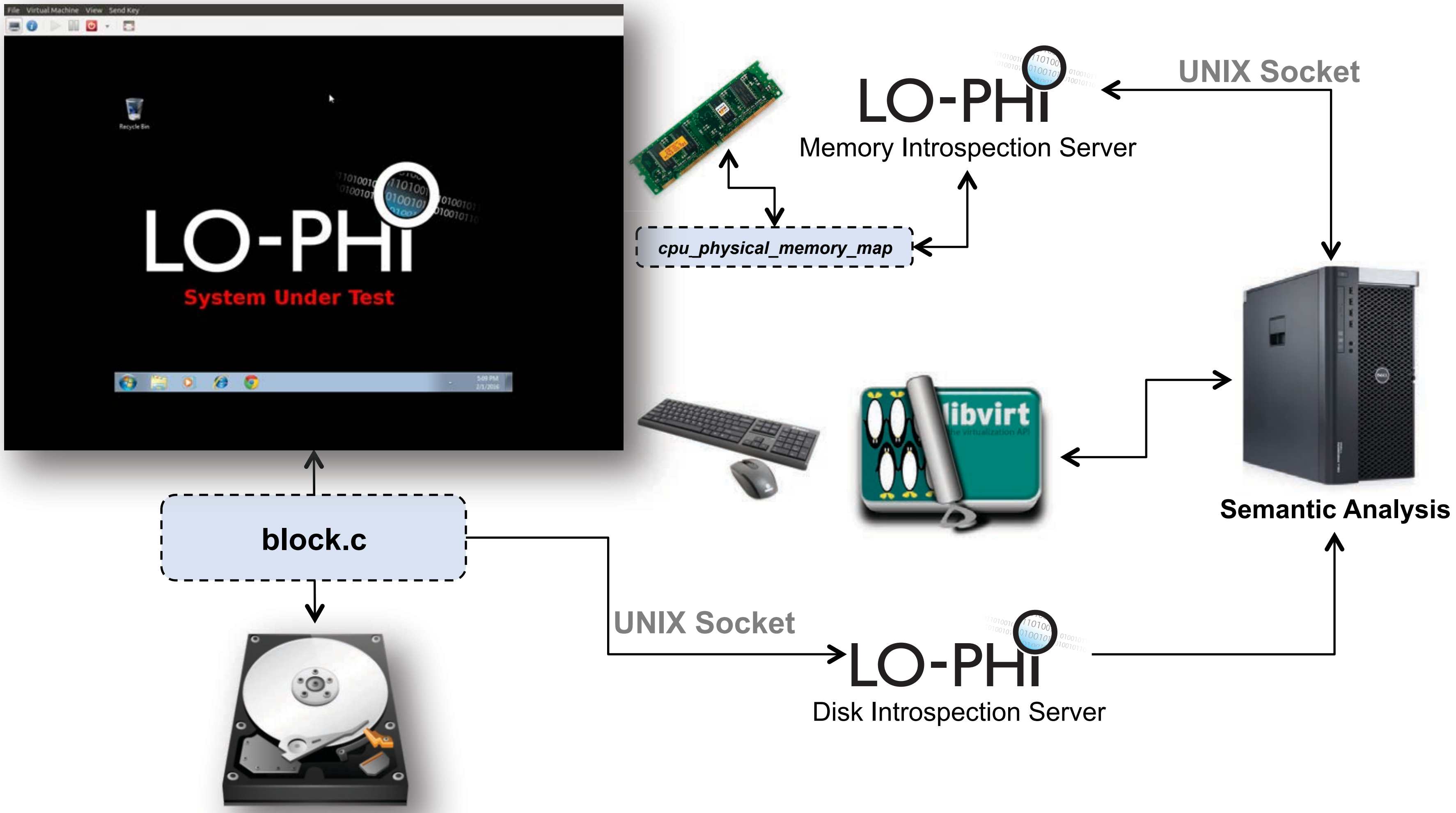


# Overview

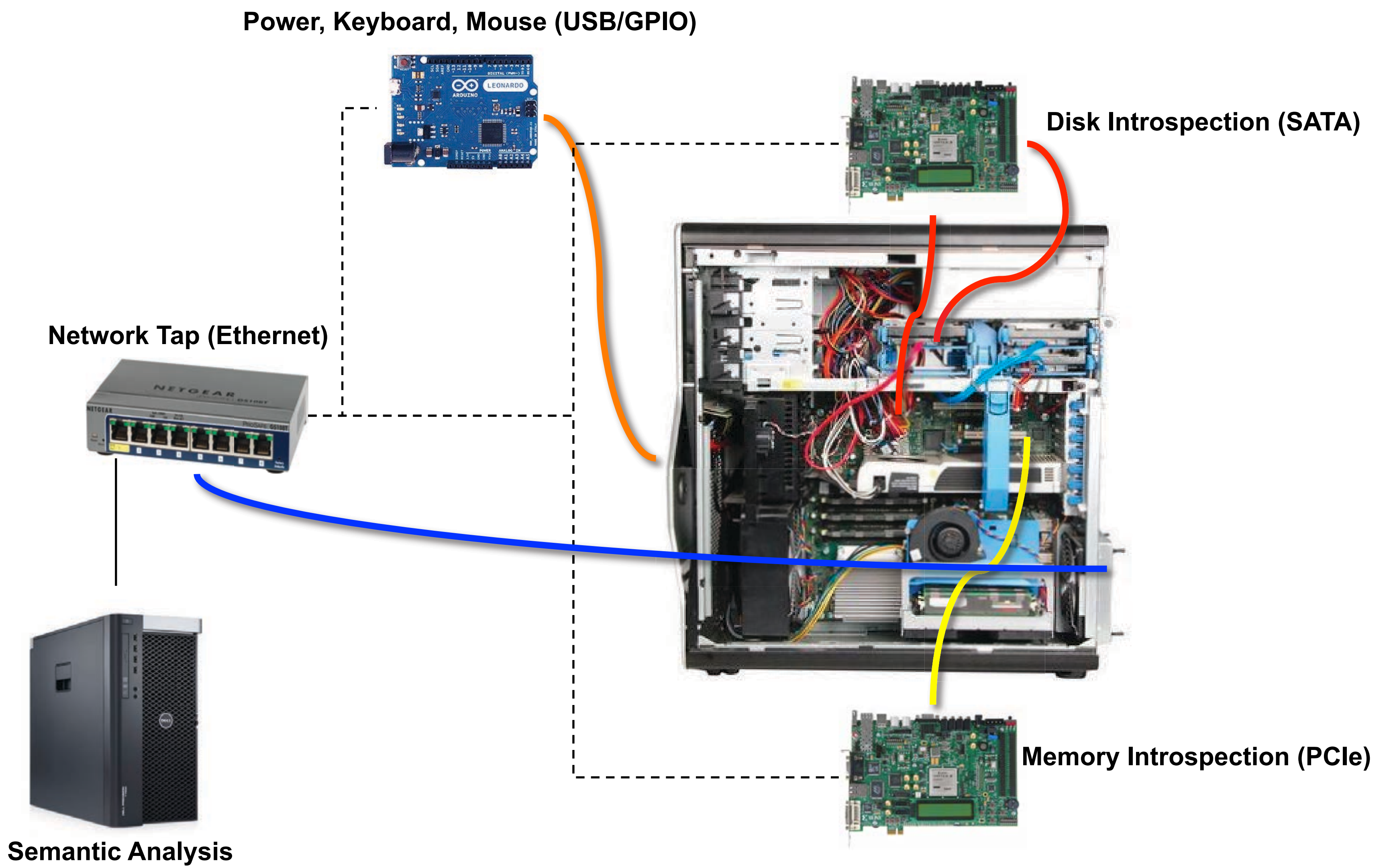
- **Zero software-based artifacts**
- **Simple Python APIs to interact with a system under test**
  - Same code for either physical or virtual machines
- **A suite of both sensors and actuators**
- **A suite of semantic-gap reconstruction tools**
- **Python-based framework for automated binary analysis**
  - Analysis “scripts” can be submitted and executed on automatically provisioned machines

# Virtual Instrumentation

LO-PHI



# Physical Instrumentation



# Semantic Gap

- **Fictional Hollywood example: *The Matrix***

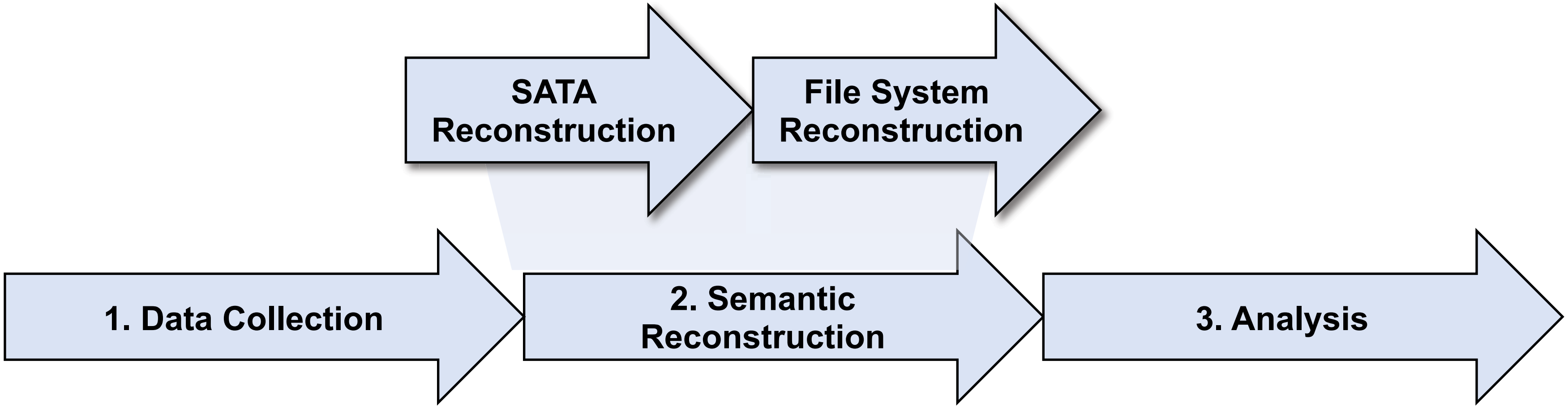
1. **Input Raw Data**
2. **Parse Data Structures**
3. **Extract Features**

- **Memory** (Volatility)
  - Reader raw memory to extract attributes of the system
    - E.g., running processes, kernel modules, descriptor tables
- **Hard Disk** (Sleuthkit)
  - Translate low-level disk activity into file system activities
    - E.g., file creation, deletion, read, write

# Stream-based Disk Forensics

## Bare Metal

- **Multiple layers of abstraction that we must bridge**
  - ✓ Analog Signal → Digital bits
  - ✓ Digital bits → SATA Frames
  } **Xilinx ML507 FPGA**
  - ✓ SATA Frames → Sector manipulation — **SATA Reconstruction**
  - ✓ Sector manipulation → File System Manipulation — **Sleuthkit (TSK) analyzeMFT**



# SATA Reconstruction

## A Brief Primer on SATA

- **Serial ATA – bus interface that replaces older IDE/ATA standards**
- **SATA uses frames (FIS) to communicate between host and device**

HIGH SPEED SERIALIZED AT ATTACHMENT
Serial ATA International Organization

| Type field value | Description |
|---|---|
| 27h | Register FIS – Host to Device |
| 34h | Register FIS – Device to Host |
| 39h | DMA Activate FIS – Device to Host |
| 41h | DMA Setup FIS – Bi-directional |
| 46h | Data FIS – Bi-directional |
| 58h | BIST Activate FIS – Bi-directional |
| 5Fh | PIO Setup FIS – Device to Host |
| A1h | Set Device Bits FIS – Device to Host |
| A6h | Reserved for future Serial ATA definition |
| B8h | Reserved for future Serial ATA definition |
| BFh | Reserved for future Serial ATA definition |
| C7h | Vendor specific |
| D4h | Vendor specific |
| D9h | Reserved for future Serial ATA definition |

**10.3.4 Register - Host to Device**

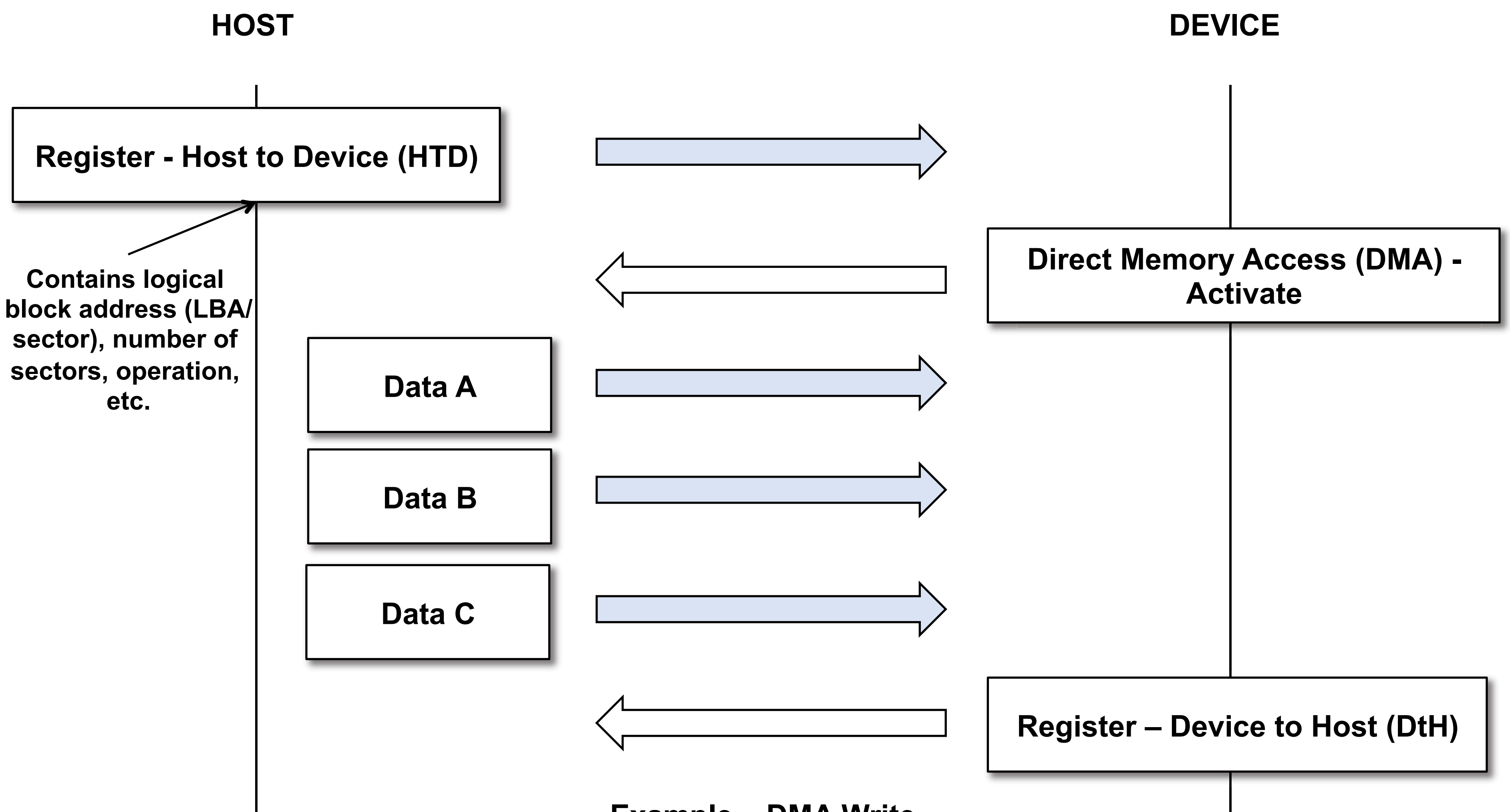
| | | | | |
|---|---|---|---|---|
| 0 | Features(7:0) | Command | C R R R PM Port | FIS Type (27h) |
| 1 | Device | LBA(23:16) | LBA(15:8) | LBA(7:0) |
| 2 | Features(15:8) | LBA(47:40) | LBA(39:32) | LBA(31:24) |
| 3 | Control | ICC (7 6 5 4 3 2 1 0) | Count(15:8) | Count(7:0) |
| 4 | Reserved (0) | Reserved (0) | Reserved (0) | Reserved (0) |

**Figure 194 – Register - Host to Device FIS layout**

FIS – Frame Information Structure

# SATA Reconstruction

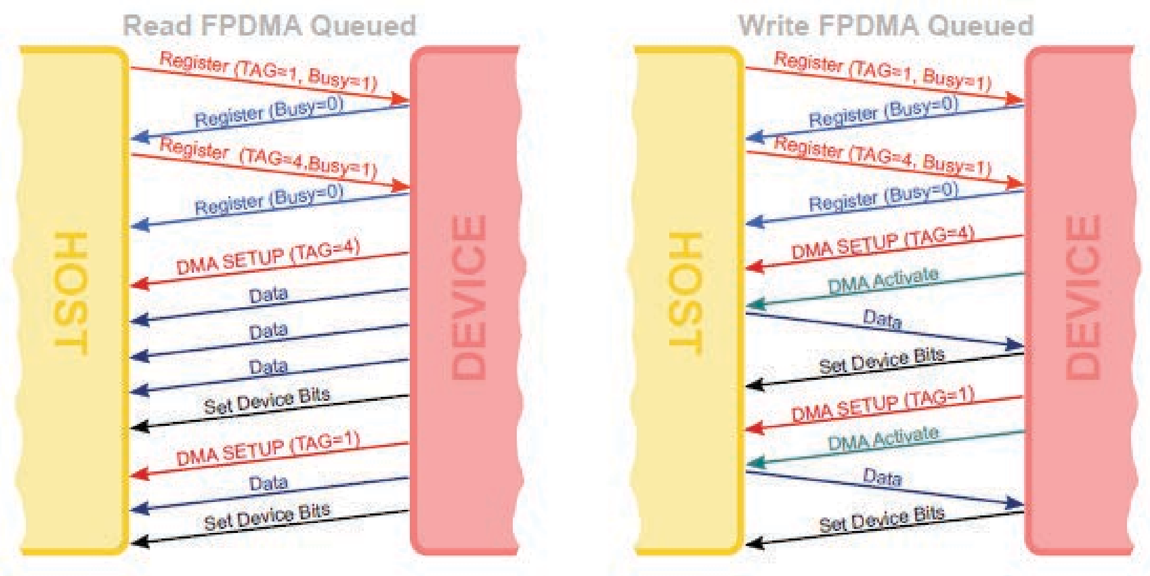
## A Brief Primer on SATA




Example – DMA Write

# SATA Reconstruction

## Native Command Queuing




- **Native Command Queuing (NCQ) complicates reconstruction**
- **NCQ allows for up to 32 separate, concurrent, asynchronous disk transactions**
  – Many SATA devices implement NCQ
- **NCQ identifies transactions by 5-bit TAG field (0-31)**

# SATA Reconstruction

LO-PHI

- **Wrote a Python module to handle all of these transactions**
  - Consumes raw SATA frames
  - Supports all of the existing SATA versions
  - Outputs stream of logical sector operations

- **Traditional SATA analyzers are expensive and don't provide analysis-friendly interfaces**

Lecroy Catalyst Stx230 2 Port **Sata** Serial Bus Protocol **Analyzer** W/
**$1,550.00 used** from eBay
Lecroy Catalyst STX230 2 Port **SATA** Serial Bus Protocol **Analyzer** Includes:• Carrying Case • USB

Finisar Xgig-C004 XGIG-C041 w/ 2X Xgig-B830Sa 8-Port SAS/**SATA** ...
**$3,995.00 used** from 2 stores

Lecroy St2-31-2a **Sata** 1.5g/3g Bus
**$4,000.00 refurbished** from eBay
LeCroy ST2-31-2A **SATA** 1.5G/3G Bus **Analyzer** Buffer Size:1GB,1port:(Host/Device),Real Time Ev
STTAP2 ...

# File System Reconstruction

- **Current Solution**
  - Uses PyTSK to keep a unified codebase in Python
  - Naïve approach requires analyzing the *entire* image at every interval

- **Optimization:** Uses AnalyzeMFT for NTFS optimization

**Extract file system state using TSK from initial *clean* image**

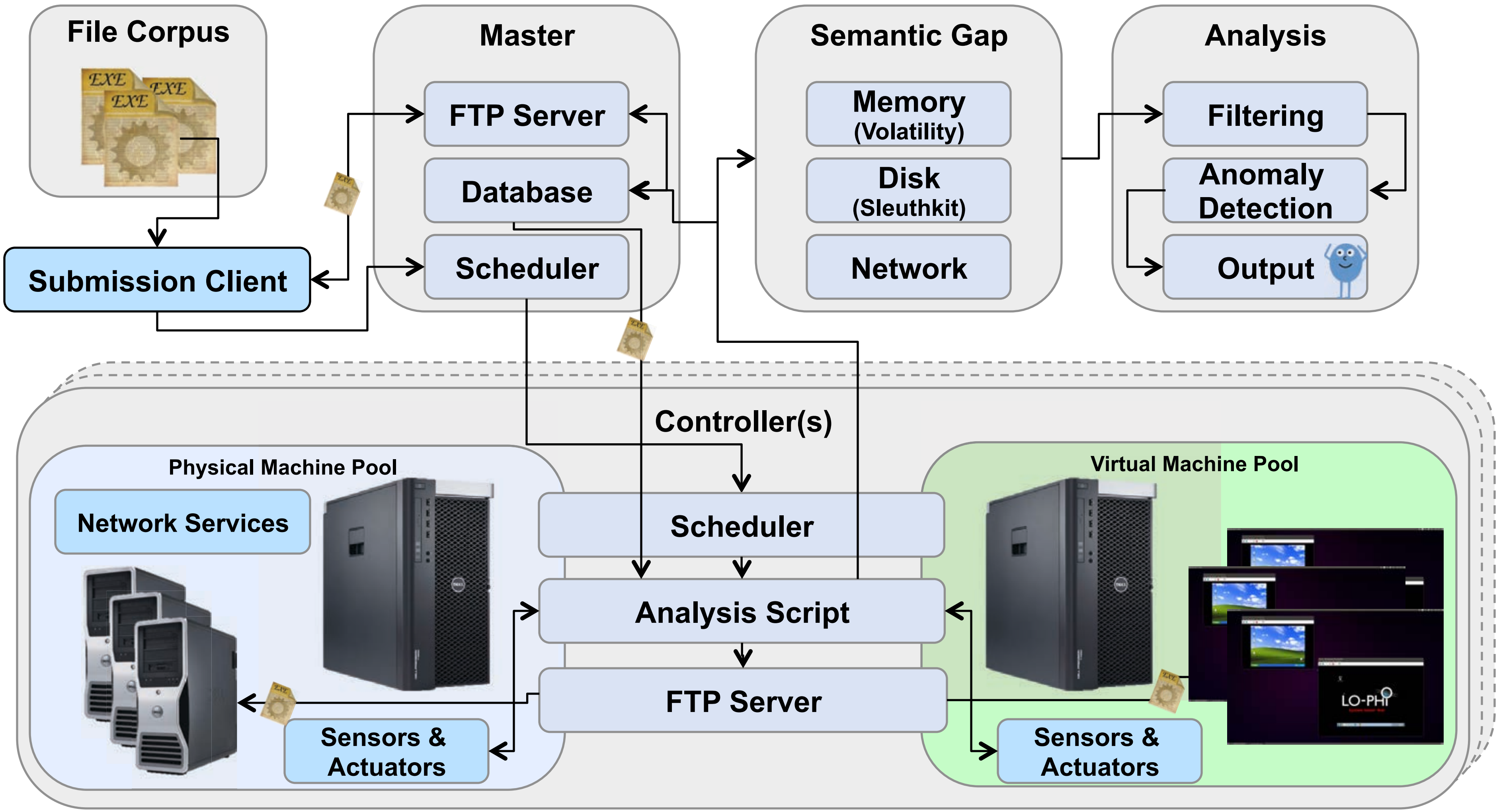
**Check previous state**

**if *known sector*: Update structures**

**else: report as UNKNOWN**

**0** **t** **t+1**

# Automated Binary Analysis



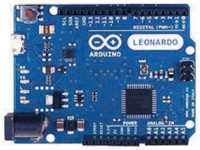
# Automated Binary Analysis

## Physical Machines

- **Machine/hard disk reset**

**1. Power down machine**

**2. Re-image disk with selected OS (CloneZilla)**



**DHCP/PXE**

**TFTP**

**DNS**

**Controller**

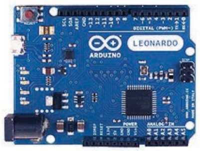
*LO-PHI Network Services*

**System Under Test**

- **Download binary onto SUT**

**3. Wait for OS to appear on the network (ping)**

**4. Download binary from controller using ftp (key presses)**



**DHCP/PXE**

**FTP**

**Controller**

*LO-PHI Network Services*

**System Under Test**

# Automated Binary Analysis

## Physical Machines

- **Execute binary**

**5. Dump clean state of memory**

**6. Start capturing network and disk activity**

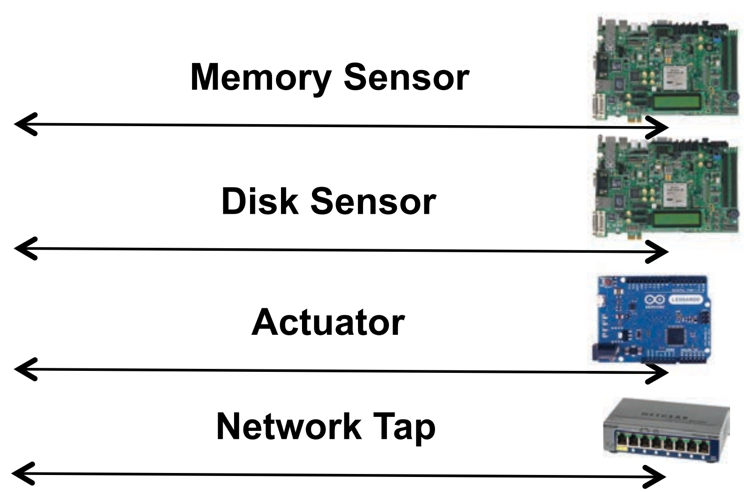
**7. Run Binary** (Start moving mouse)

**8. Dump interim state of memory**

**7. Identify and click all buttons** (Volatility)

**8. Dump dirty state of memory**

**Controller**




**System Under Test**

# Evaluation: Semantic Output

(on WinXPSP3)

- **Homemade Rootkit**
  - **Comparison:** Anubis failed to execute the binary, and Cuckoo sandbox failed to detect/execute our ftp server
- **Labeled Malware** *(213 well-labeled samples)*
  - Blind analysis identified various behaviors, all of which were confirmed by ground truth
- **Unlabeled Malware** *(1091 samples)*
  - Similar findings

| Observed Behavior | Number of Samples |
|---|---|
| Created new process(es) | 765 |
| Opened socket(s) | 210 |
| Started service(s) | 300 |
| Loaded kernel modules | 20 |
| Modified GDT | 58 |
| Modified IDT | 10 |

# Evaluation: Evasive Malware

(on Windows 7)

- **Paranoid Fish** *(Evasive malware proof-of-concept)*
  - Failed to detect LO-PHI
  - **Comparison:** Anubis and Cuckoo sandbox were both detected due to virtualization artifacts

- **Labeled Malware** (429 coarsely-labeled samples)
  - LO-PHI detected *suspicious activity* in almost every sample
    - Some appeared to be targeting a different OS version

| Technique Employed | # Samples |
|---|---|
| Wait for keyboard | 3 |
| Bios-based | 6 |
| Hardware id-based | 28 |
| Processor feature-based | 62 |
| Exception-based | 79 |
| Timing-based | 251 |

| Malware Label | Volatility Module: envars | netscan | ldrmodules | psxview | buttons |
|---|---|---|---|---|---|
| Keyboard | 0 | 3 | 1 | 0 | 1 |
| Bios | 3 | 6 | 6 | 6 | 0 |
| Hardware | 28 | 27 | 28 | 26 | 11 |
| Processor | 53 | 54 | 59 | 51 | 7 |
| Exception | 76 | 79 | 77 | 76 | 7 |
| Timing | 229 | 247 | 231 | 239 | 4 |

# Summary

- **Deployed and tested LO-PHI an extremely low-artifact, hardware and VM-based, dynamic-analysis environment**

- **Developed hardware, and supporting tools, for stream-based disk forensics on SATA-based physical machines[1]**

- **Constructed a framework, and accompanying infrastructure, for automating analysis of binaries on both physical and virtual machines**
  - Open Source (BSD License): http://github.com/mit-ll/LO-PHI

- **Demonstrated the scalability and fidelity of LO-PHI by analyzing thousands of labeled and unlabeled malware samples**

[1]http://www.osdfcon.org/presentations/2014/Hu-Spensky-OSDFCon2014.pdf

# Demo

Demonstration of VM-based binary analysis.