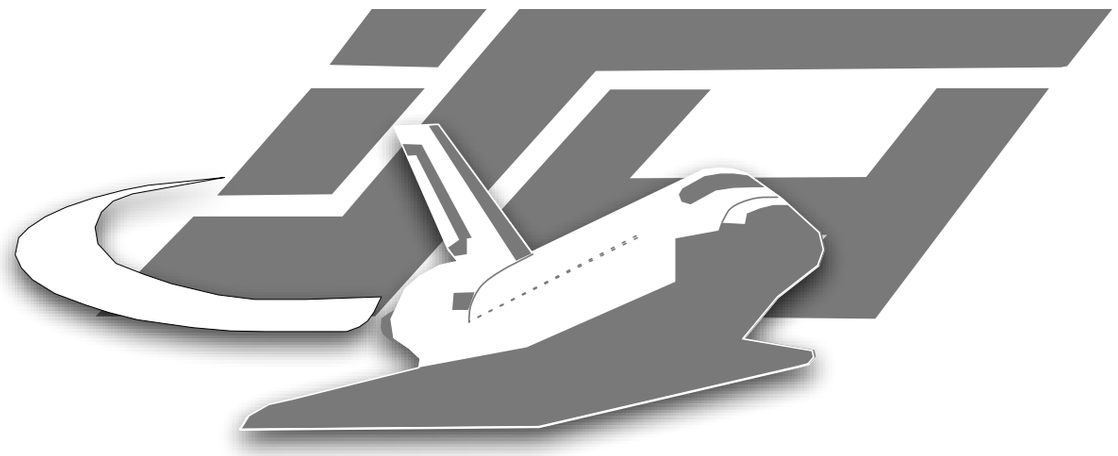
FireWire

all your memory are belong to us

Michael Becher, Maximillian Dornseif, Christian N. Klein

http://md.hudora.de/presentations/#firewire-cansecwest

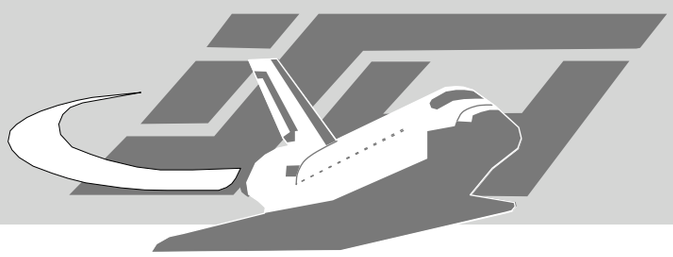


Laboratory for Dependable Distributed Systems

RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

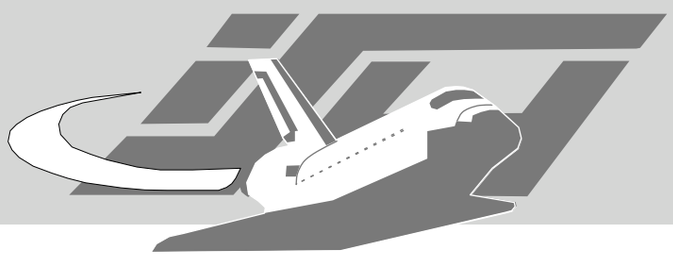
RWTH

UNIVERSITY
OF TECHNOLOGY
AACHEN



Agenda

- Who we are and what we do
- Introduction to FireWire
- Technical Details of FireWire
- Demo
- Implementation Details
- Forensics by FireWire
- What to do about the issue

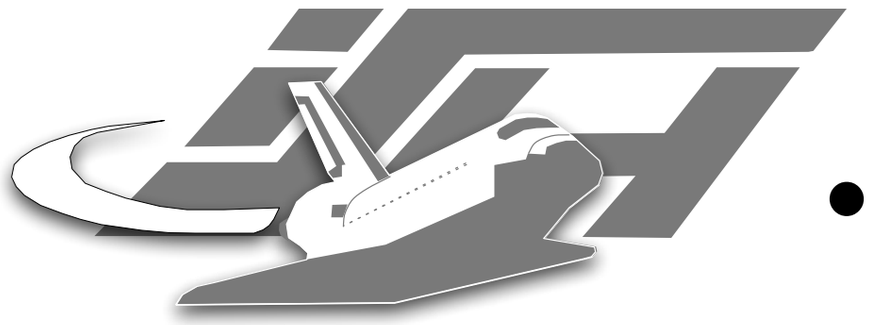


Who we are

RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

RWTH

UNIVERSITY
OF TECHNOLOGY
AACHEN



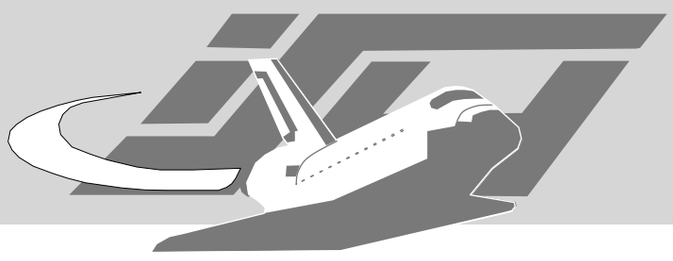
Laboratory for Dependable Distributed Systems

- Laboratory for Dependable Distributed Systems at RWTH-Aachen University
  - Founded in late 2003 for theoretical & practical security research, topics include:
    - Security Education
    - Sensor Networks
    - Honeypot technology
    - Breaking Stuff
  - Notable classes include “Hacker Seminar”, “Hacker Praktikum”, “Pen-Test Praktikum”, “Aachen Summerschool applied IT-Security”, “Computer Forensics” ... and the “RedTeam”
  - http://mail-i4.informatik.rwth-aachen.de/mailman/listinfo/lufgtalk/

Introduction into FireWire

What is Firewire?

- Developed by Apple Computers since 1985
- IEEE 1394 (1995), IEEE 1394a (2000), IEEE 1394b (2002).
- Marketed by Apple as “Firewire” or “FireWire”
- Marketed by Sony as “iLink”



FireWire

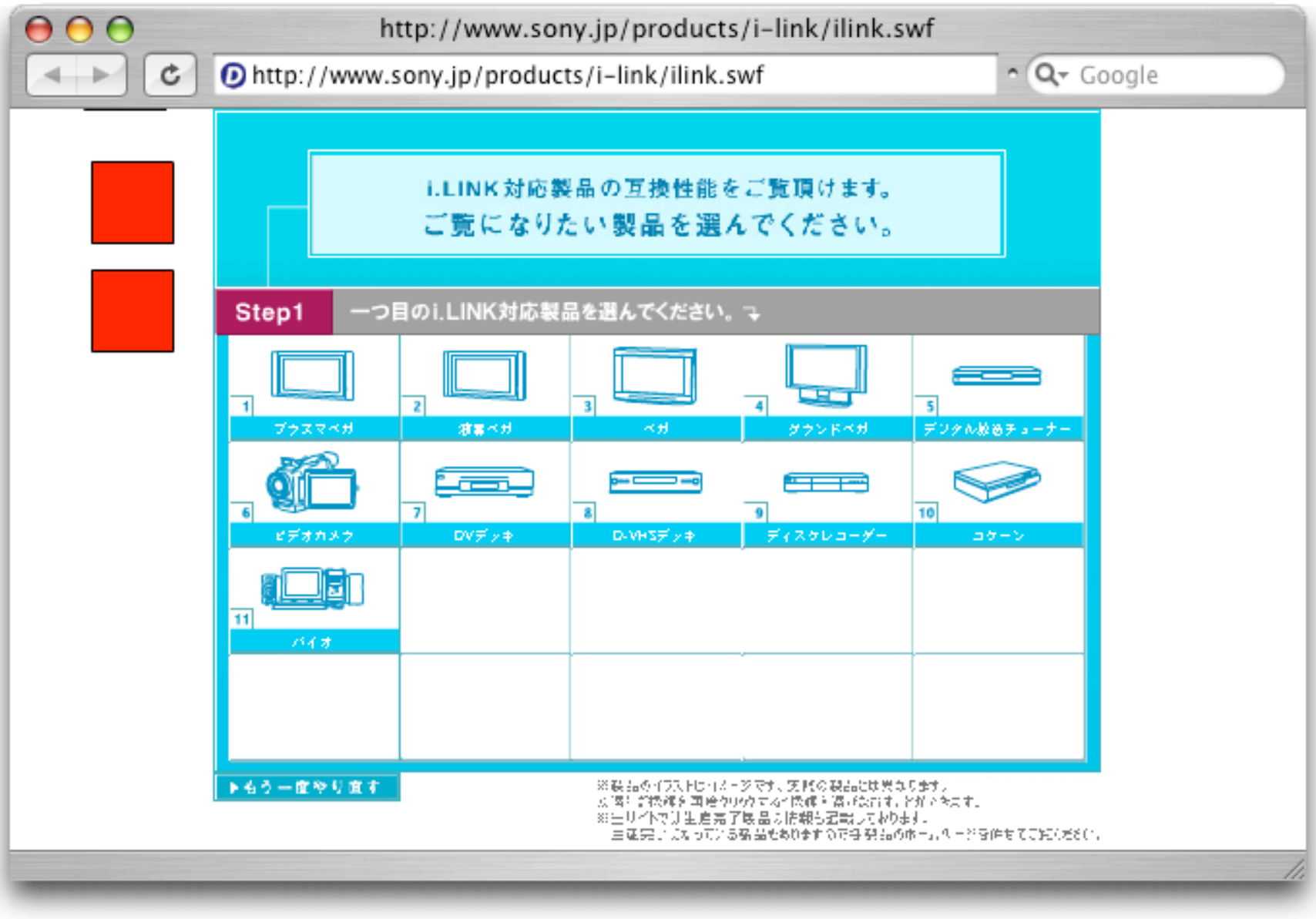
- Serial bus, similar but more sophisticated than USB
- Faster
- Peer-to-Peer, needs no computer
- More Power
  - in many respects

Marketplace

- Apple - pushing FireWire hard:
  - Since January 1999 in Desktops
  - Since January 2000 in Notebooks
  - September 2000 where the last non-FireWire machines shipped
  - October 2001: iPod as FireWire killer-app

- Sony - we'll come to that
- Others: most upper class systems come with FireWire

FireWire by Sony

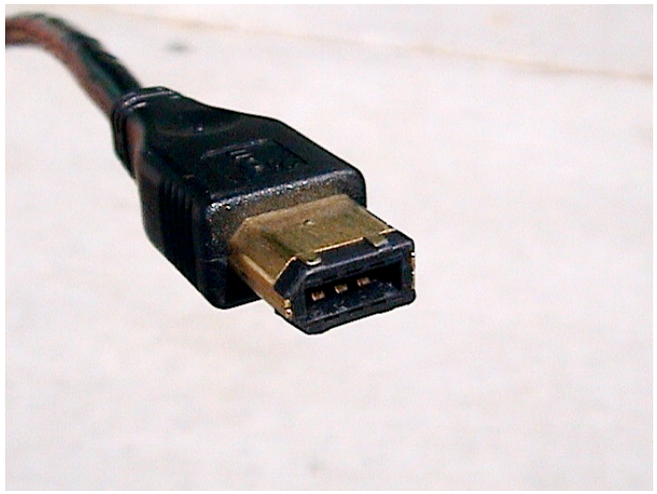


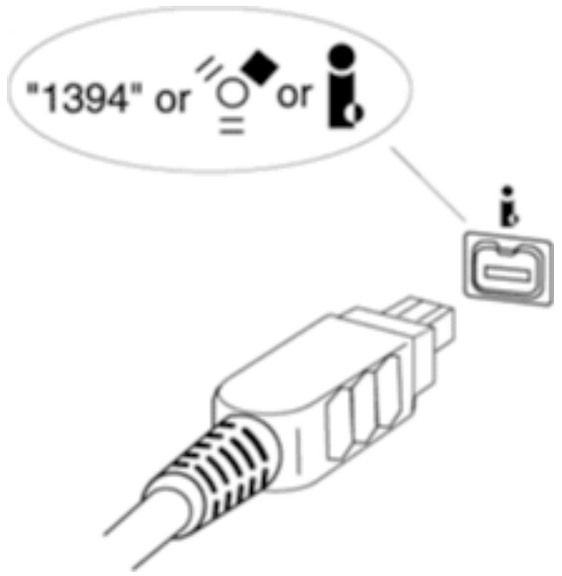
More FireWire

- Audio
- Printers
- Scanners
- Cameras
- GPS
- Lab Equipment
- Industrial Control

Things to come

Confusion





Interesting use: Target Disk Mode

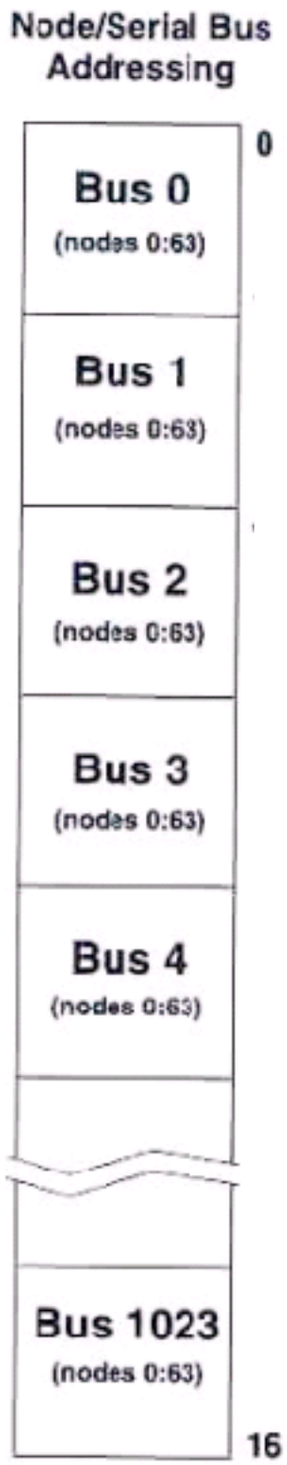
- Press “T” while powering up
- Macintosh will emulate a Firewire disk drive

Technical Details of FireWire/OHCI

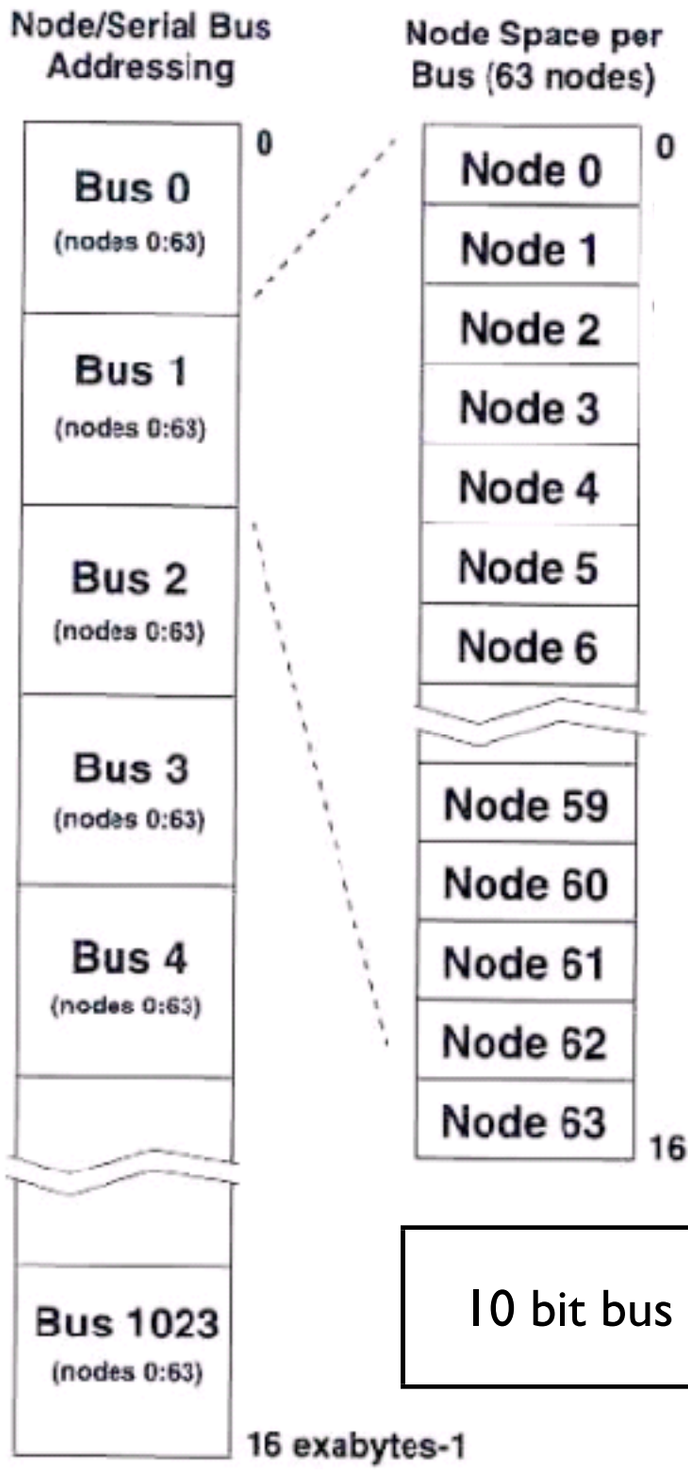
64 bit unified Memory space: 10 bit bus ID, 6 bit node ID, 48 bit per node

10 bit bus ID	6 bit node ID	48 bit mem location

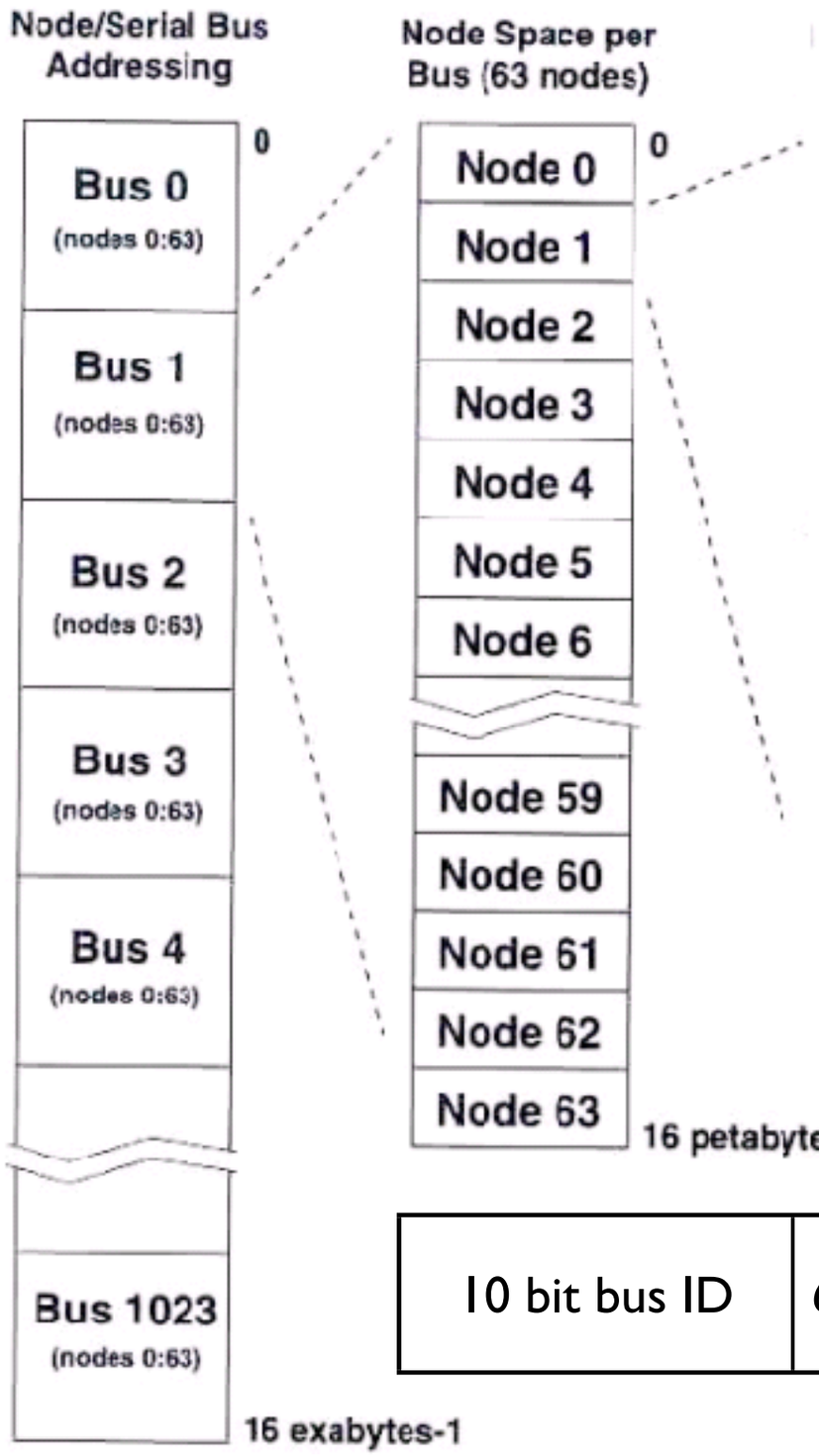
64 bit unified Memory space: 10 bit bus ID, 6 bit node ID, 48 bit per node

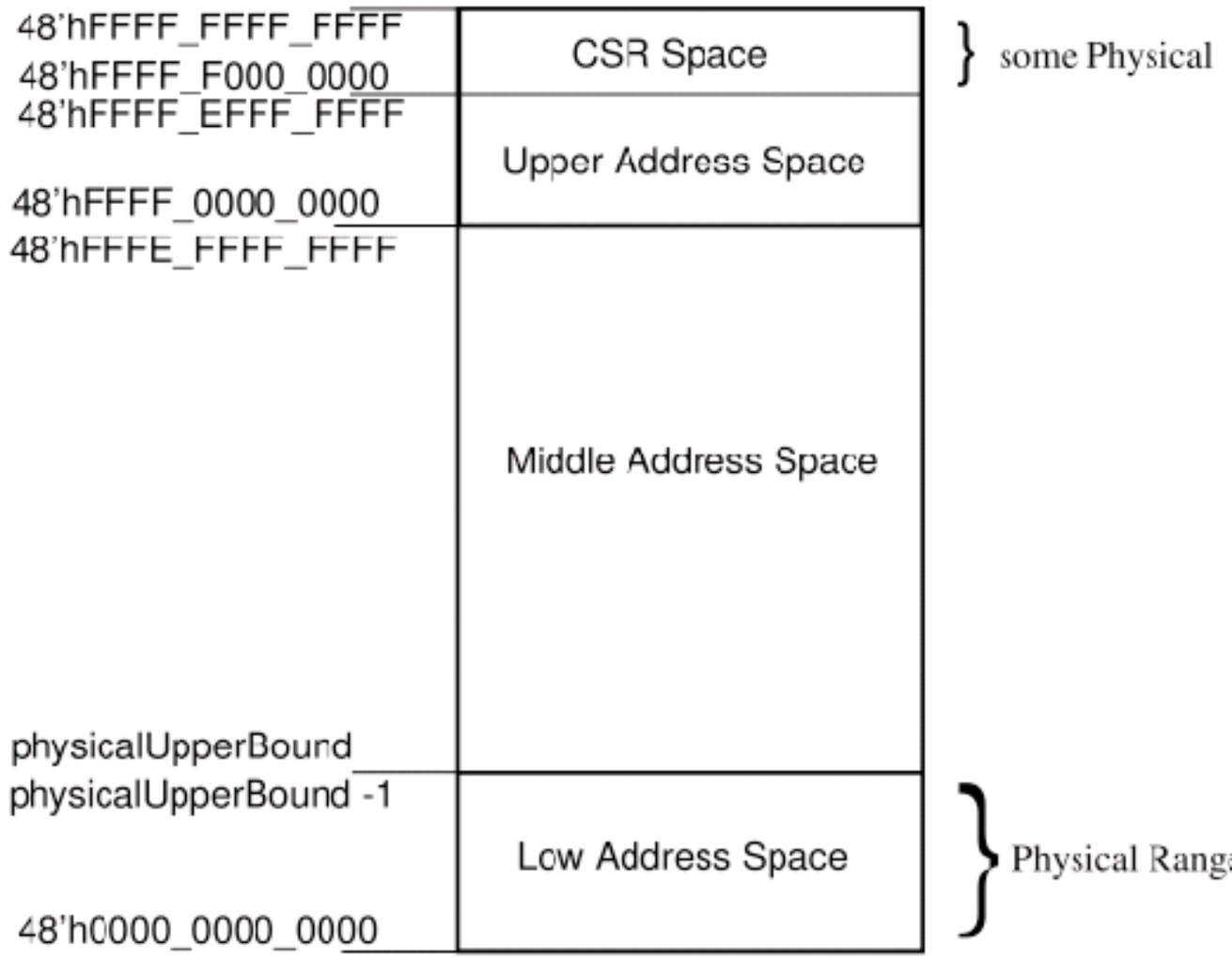


64 bit unified Memory space: 10 bit bus ID, 6 bit node ID, 48 bit per node



64 bit unified Memory space: 10 bit bus ID, 6 bit node ID, 48 bit per node





10 bit bus ID	6 bit node ID	48 bit mem location

OHCI

- Asynchronous functions
  - Can be used to access on-board RAM and RAM on extension cards (PCI)

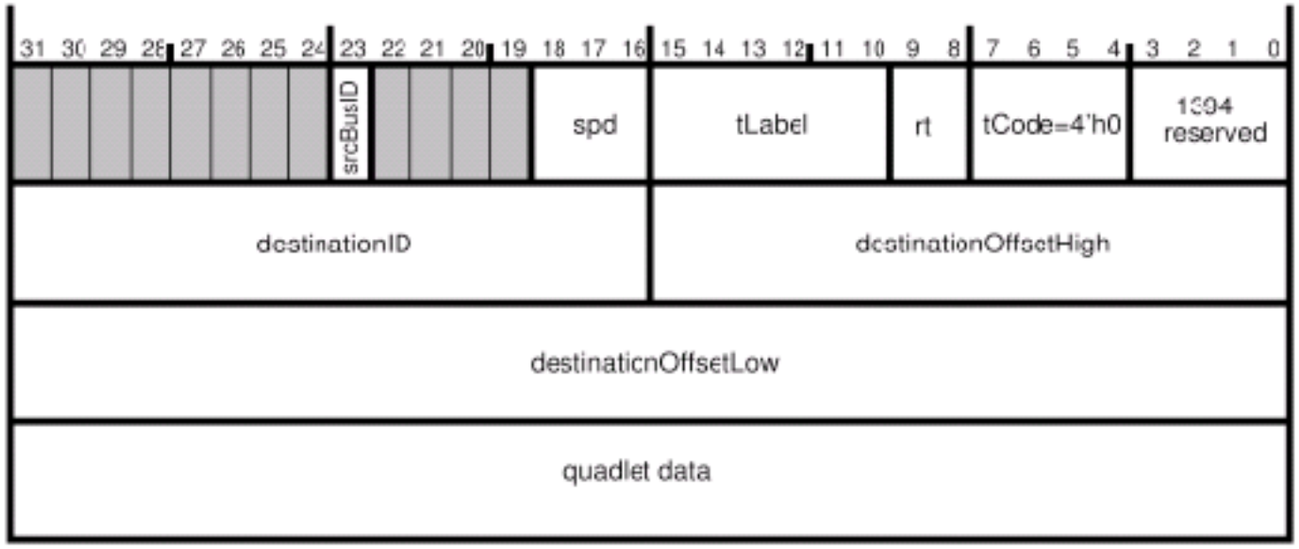
"physical requests, including physical read, physical write and lock requests to some CSR registers (section 5.5), are handled directly by the Host Controller without assistance by system software." (OHCI Standard)

OHCI Filters

- "Asynchronous Request Filters"
  "The 1394 Open HCI allows for selective access to host memory and the Asynchronous Receive Request context so that software can maintain host memory integrity. The selective access is provided by two sets of 64-bit registers: PhysRequestFilter and AsynchRequestFilter. These registers allow access to physical memory and the AR Request context on a nodeID basis." (OHCI Standard)

- PhysicalRequestFilter Registers (set and clear)
  "If an asynchronous request is received, passes the AsynchronousRequestFilter, and the offset is below PhysicalUpper-Bound (section 5.15), the sourceID of the request is used as an index into the PhysicalRequestFilter. If the corresponding bit in the PhysicalRequestFilter is set to 0, then the request shall be forwarded to the Asynchronous Receive Request DMA context. If however, the bit is set to 1, then the request shall be sent to the physical response unit." (OHCI Standard)

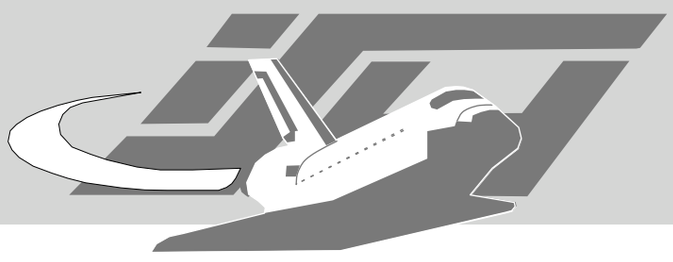
- quartlet write request transmit:



- for details see:

Exploiting Reads

- We can read arbitrary memory locations.
  So we can:
  - Grab the Screen contents
  - Just search the memory for strings
  - Scan for possible key material
    - “Playing hide and seek with stored keys” by Someren/Shamir 1998
      http://www.ncipher.com/resources/downloads/files/white_papers/keyhide2.pdf
  - Parse the whole physical memory to understand logical memory layout.



Exploiting Writes

- We can write arbitrary data to arbitrary memory location. So we can:
  - Mess up
  - Change screen content
  - Change UID/GID of a certain process
  - Inject code into a process
  - Inject an additional Process

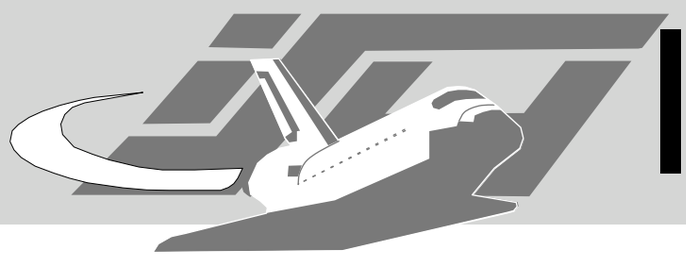
Macintosh HD

Demo

Maximillian Dornseif • Laboratory for Dependable Distributed Systems

RWTH AACHEN

Implementation Details



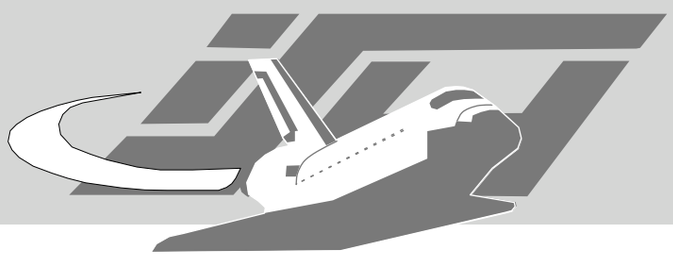
Implementation

- MacOS X (IOFireWire* Frameworks)

```
IOCreatePluginInterfaceForService(self->aDevice, kIOFireWireLibTypeID,
kIOCFPlugInInterfaceID, &cfPlugInInterface, &theScore);
(*cfPlugInInterface)->QueryInterface(cfPlugInInterface,
CFUUIDGetUUIDBytes(kIOFireWireDeviceInterfaceID), (void **)&fwIntf);
(*fwIntf)->Open(fwIntf);
(*fwIntf)->Write(fwIntf, self->aDevice, &fwaddr, (void *) buffer, &bufsize, false, 0);
(*fwIntf)->Read(fwIntf, self->aDevice, &fwaddr, (void *) buffer, &bufsize, false, 0);
```

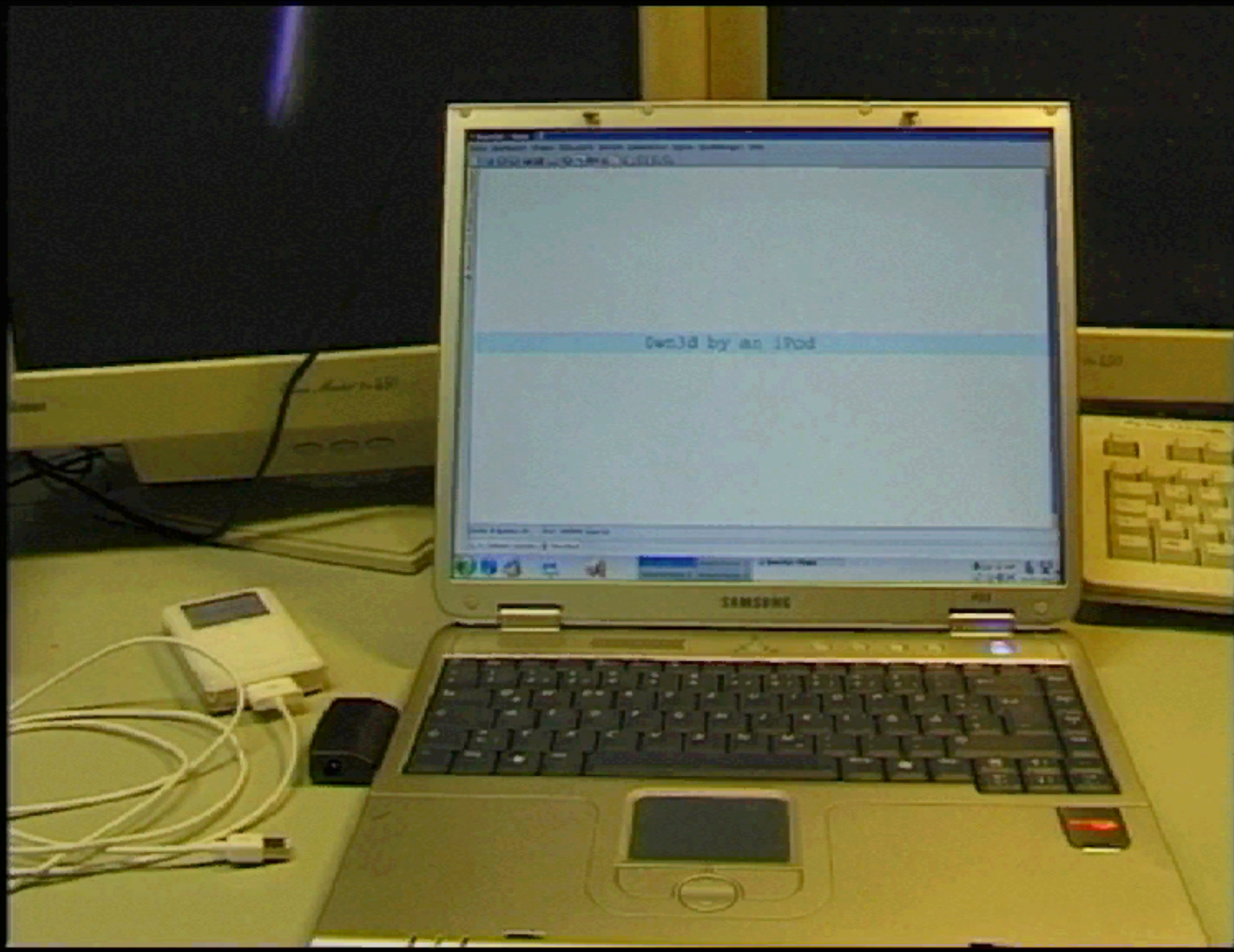
- Linux (libraw1394)

```
handle = raw1394_new_handle();
raw1394_set_port(handle, 0);
raw1394_write(handle, node_id, fwaddr, bufsize, (quadlet_t *) buf);
```



pyfw

- python warper around native FireWire APIs
- used for all our demos
- ported since 2004-05-04 08:45 to Linux and iPod Linux
- Get it at http://md.hudora.de/presentations/#firewire-cansecwest


Owned by an iPod
SAMSUNG

Macintosh HD

Demo

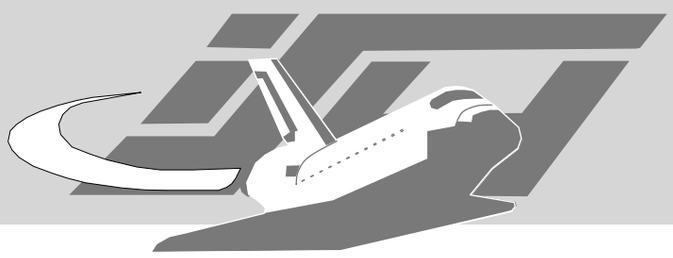
Maximillian Dornseif • Laboratory for Dependable Distributed Systems

RWTH AACHEN

	read	write
MacOS	works	works
FreeBSD	works	works
Linux	nope	works
Windows 2000	CRASH	CRASH
Windows XP	nope	nope

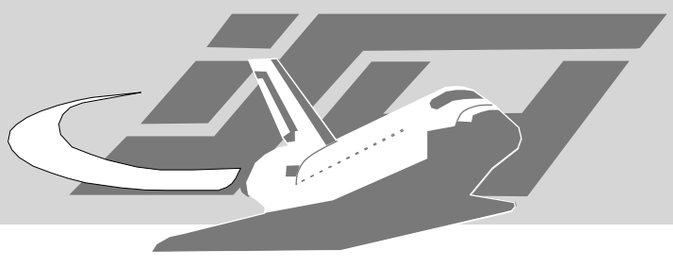
	read	write
MacOS 10.3.9	nope	nope
FreeBSD	works	works
Linux	nope	works
Windows 2000	?	?
Windows XP	nope	nope

Forensics by Firewire



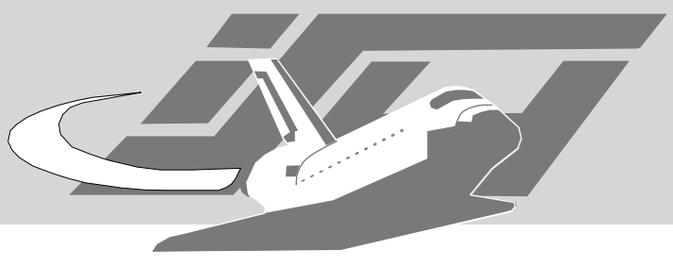
The forensics schism

- Unplug, do post-mortem disk-analysis
  - Misses Processes, open connections, etc.
- Gather information on the live system, afterwards do a clean shutdown and do afterwards disk-analysis
  - Contaminates evidence during the information gathering



Live Memory Dumps

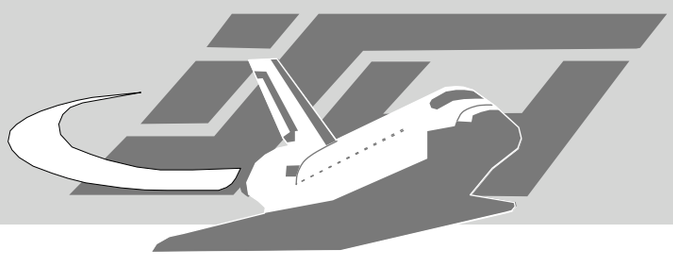
- Being able to dump the whole memory without software support would solve the schism
- Tribble is a specialized pice of hardware being able to dump physical memory via DMA transfers over the PCI bus
- If you can do the same via Firewire, you get away with a software only solution



Forensics Challenges

- There is little experience in reconstructing logical/virtual memory from physical memory dumps
- To find open network connections etc. we have to parse a bunch of kernel structures

Defenses



Shields-Up!

- Ensure that only fully trusted devices are connected to your FireWire ports
- Press you driver/OS vendors about FireWire filtering

Filtering: Linux

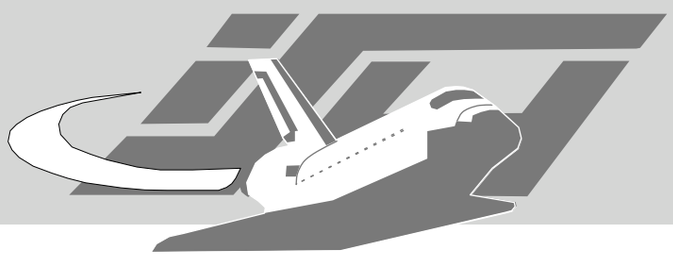
Terminal – RedTeam@RWTH

ohci1394.c

```
/* Accept Physical requests from all nodes. */
reg_write(ohci,OHCI1394_AsReqFilterHiSet, 0xffffffff);
reg_write(ohci,OHCI1394_AsReqFilterLoSet, 0xffffffff);
/* Turn on phys dma reception.
 *
 * TODO: Enable some sort of filtering management.
 */
if (phys_dma) {
  reg_write(ohci,OHCI1394_PhyReqFilterHiSet, 0xffffffff);
  reg_write(ohci,OHCI1394_PhyReqFilterLoSet, 0xffffffff);
  reg_write(ohci,OHCI1394_PhyUpperBound, 0xffff0000);
} else {
  reg_write(ohci,OHCI1394_PhyReqFilterHiSet, 0x00000000);
  reg_write(ohci,OHCI1394_PhyReqFilterLoSet, 0x00000000);
}
DBGMSG("PhyReqFilter=%08x%08x",
  reg_read(ohci,OHCI1394_PhyReqFilterHiSet),
  reg_read(ohci,OHCI1394_PhyReqFilterLoSet));
```

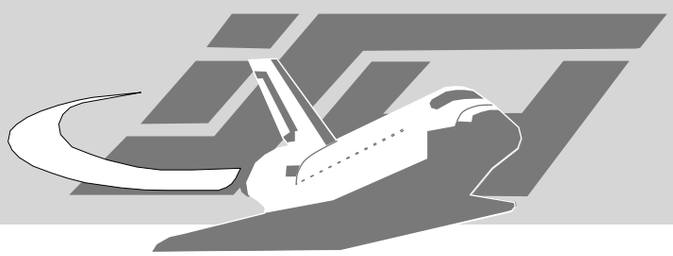
Filtering: MacOS

```
IOFireWireController.cpp
IOFWSecurityMode mode = kIOFWSecurityModeNormal;
OSString * securityModeProperty = OSDynamicCast( \
OSString,options->getProperty("security-mode") );
if(securityModeProperty != NULL &&
   strcmp("none", securityModeProperty->getCStringNoCopy()) != 0)
{
  // set security mode to secure/permanent
  mode = kIOFWSecurityModeSecurePermanent;
}

[...]

// shut them all down!
fFWIM->setNodeIDPhysicalFilter( kIOFWAllPhysicalFilters, false );
```



Be Prepared for Forensics

- You might want to keep FireWire ports on incident prone systems at hand
  - Keep them physically secured
- Have some software ready to do memory dumps via FireWire



- References
  - http://md.hudora.de/presentations/#firewire-cansecwest
    - updated slides
    - pyfw MacOS/Linux code
    - Videos of the Demos
    - Within a few days:
      - Linux image for the iPod with all the goodies

    http://mail-i4.informatik.rwth-aachen.de/mailman/listinfo/lufgtalk/
- Thanks
  - Christian N Klein for the initial MacOS code
  - Michael Becher for the Linux/iPod code