What's on the Wire? Physical Layer Tapping with Project Daisho

Dominic Spill, Michael Kershaw, Michael Ossmann

Black Hat USA 2013

Abstract

Daisho is a project to produce an extensible, open source monitor for high speed communication media. Using a capable Field-Programmable Gate Array (FPGA) and off-the-shelf transceiver components, we are able to monitor communication at the lowest possible layer, providing users with the data needed to analyze any higher layer protocols that are encapsulated.

In order to integrate with current workflows, Daisho has been designed to work with, and build upon, existing open tools and standards, such as Wireshark and the pcap file format.

The platform offers the ability to monitor the current generation of communication technologies and provides a base from which to target future technologies.

1 Introduction

Daisho[1] is a project to produce an extensible, open source monitor for wired communication media. A survey of the current crop of technologies reveals a lack of network tapping solutions that provide access at the lowest possible layer of a communication link.

We believe that physical layer network taps are crucial tools to ensure the security of our communications; specifically we subscribe to a belief succinctly stated by Joshua Wright as "*Security will not get better until tools for practical exploration of the attack surface are made available.*"[2]

Unless tools exist to explore the security of technologies that we rely on to communicate, we cannot have any faith in the security of these technologies. For example, early 802.11 chipsets supported a monitor mode which allowed researchers to discover vulnerabilities in both WEP and WPA security measures, forcing standards organizations and manufacturers to specify and implement better authentication and encryption methods.

Many technologies that we use today do not have low level monitors that are available to security researchers, and it is therefore not possible to trust that the implementations of these protocols are secure and standards compliant.

1.1 Background

The idea for Daisho was born out of the shortcomings of the current generation of network monitoring tools. Devices such as the Throwing Star LAN Tap[3] are able to monitor full-duplex 100BASE-TX connections but are unable to monitor 1000BASE-T connections. Ethernet monitoring tools that support 1000BASE-T do not provide visibility below the Media Access Control (MAC) layer.

Tools such as usbmon[4] can be used to monitor communication between a USB host and device but operate only on the data within the host system, so they cannot be used to validate a USB host implementation's transmissions on the wire. Such tools are also difficult to use for reverse engineering proprietary protocols as they are only supported on certain operating systems, making it impossible to monitor the communications between an unsupported host and an unsupported device. For example, development of the Open Kinect driver required an external USB analyzer[5].

Initially we considered the feasibility of building a microcontroller-based USB man-in-the-middle platform allowing users to tap and/or modify data between a USB host and device. A platform of this nature would also allow us to experiment with novel bus topologies, such as connecting two USB devices together, for example connecting a USB keyboard to a Bluetooth adapter to produce a Bluetooth keyboard or cloning mass storage devices without the need for a host system.

We found it difficult to design a microcontroller-based solution that would accomplish our goals. Specifically we wanted the platform to support Hi-Speed USB in On-The-Go (OTG) mode on two target ports while simultaneously transferring monitored data over a third Hi-Speed USB connection to an attached host computer. This difficulty, along with the desire to produce a modular platform supporting multiple target communication technologies, led us to explore an architecture employing programmable logic.

2 Architecture

The Daisho platform is a device designed to attach to a back-end host computer over a SuperSpeed USB 3.0 connection. It is composed of two parts. The primary part is a mainboard including the back-end USB connection and a Field-Programmable Gate Array (FPGA) for implementing monitoring logic. The secondary part is an interchangeable front-end module consisting of physical layer transceivers and connectors for a particular wired communication medium.

One of the key techniques employed in the design of Daisho involves the simplicity with which we are able to capture data at the lowest possible layer. Rather than designing complex and custom circuits to extract data from a communications channel, we use off-the-shelf parts as found in the devices on either end of the connection. For Ethernet these parts are physical layer transceivers (PHYs) which are a component of every Ethernet card or chipset available. For USB 2.0 and 3.0 the parts are UTMI+ Low Pin Interface (ULPI) and PIPE3 PHY ICs respectively. By using standard parts we are able to cheaply and easily convert the physical layer signals into data that can be manipulated within an FPGA and copied to a host monitoring station.

Instead of using a single receive-only circuit and passively tapping the target communication medium, we use a pair of these transceivers, inserting the Daisho platform into the middle of the target link. This man-in-the-middle architecture enables low level monitoring of technologies such as 1000BASE-T that are difficult to passively tap. It also enables future capabilities beyond monitoring. These include injection or modification of data on the target medium, opening the door to a wide variety of communication security research.

3 Design

3.1 Mainboard

The primary component of the Daisho mainboard is the FPGA that processes all data passing between front-end PHYs, as shown in Figure 1. It also con-

trols USB communication with the host computer and enables customization for future extensions.

The mainboard features a SuperSpeed USB 3.0 transceiver and micro-AB connector. The transceiver is is controlled by a USB 3.0 device core of our own design operating on the FPGA. As far as we know, this is the first open source USB 3.0 core.

A connector carrying approximately 150 data signals allows the connection of a front-end module to the mainboard.

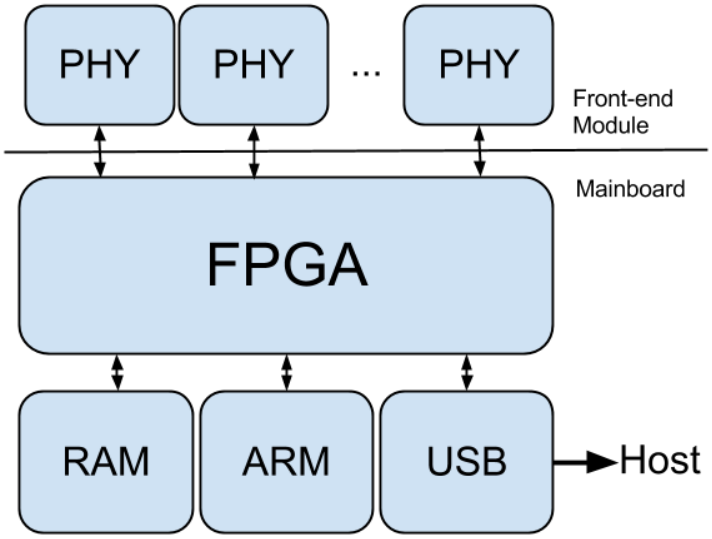

Figure 1: a high level view of the Daisho architecture

3.2 Front-end Modules

The main goal of the project is to build a platform on top of which future network monitors can be built. Additionally we have chosen to focus on 1000BASE-T Ethernet, USB 3.0 and HDMI as we believe that these technologies are both prevalent in the current crop of consumer devices and lacking in low level monitoring tools, making them ideal targets for exploration.

These technologies cover a wide range of devices, from personal computers to home entertainment systems, phones to mass storage devices and a vast array of embedded systems.

In addition to these three technologies we have chosen to build an RS-232 tap

as a proof of concept for our design and because the protocol is widely used to communicate with embedded systems.

3.2.1 RS-232

RS-232 serial is still in use today to interface with assorted hardware, such as the console of routers, switches, and servers, as well as various sensors, payment processing systems, and other devices which still employ modem technology. As such it is worth being able to accurately monitor the serial communications.

While many devices only utilize RS-232's data transmit and receive signals, RS-232 also control includes control signals for carrier detection, duplex control, and other purposes. We consider it important to provide a log of all data including these control signals.

RS-232 uses high-voltage (plus and minus 12 volt) signaling which must be adapted to TTL voltage levels for integration with the FPGA. Fortunately, there are a multitude of chips designed for exactly this purpose.

The Daisho RS-232 front-end module contains a pair of interfaces, Data Terminal Equipment (DTE) and Data Communications Equipment (DCE), which allow it to be placed in-line on a target connection. Each incoming signal is converted to TTL levels, passed to the FPGA for processing and then converted back to RS-232 voltage levels for transmission onto the opposite target interface.

3.2.2 1000BASE-T

The 1000BASE-T front-end module was one of the original inspirations of the Daisho project. The 10BASE-T and 100BASE-TX Ethernet standards divide the transmit and receive signals among discrete pairs of wires in the cable, allowing for the creation of a device for completely passive monitoring of each half of the conversation[3]. The 1000BASE-T standard, on the other hand, implements transmit and receive simultaneously across all four pairs in the cable, making it impossible to monitor with an unpowered device.

Additionally, while there are many 1000BASE-T taps on the market, to the best of our knowledge none are non-invasive, monitoring the physical layer without affecting the MAC layer. Every existing tap implements both PHY and MAC layers, appearing on the network as an Ethernet device. Mirror or "SPAN" ports on managed Ethernet switches require the switch to be part of the Ethernet network, as do low-cost taps based on Ethernet switch ICs.

The Daisho 1000BASE-T module utilizes two Gigabit Ethernet PHY chips but has no MAC. Packets from either PHY interface are passed to the FPGA and re-emitted via the other port. This allows us to be completely invisible at the MAC layer, and performing the packet copy between interfaces on the FPGA

without a MAC implementation minimizes jitter or latency introduced by our platform.

The design also allows us to timestamp packets at the FPGA, allowing for extremely high precision time records. This enables analysis of server and application latency as well timing-based side channels.

The Ethernet front-end module implements all required jack magnetics permitting Power over Ethernet pass-through.

3.2.3 USB 3.0

The front-end module targeting USB 3.0 is similar in design to the 1000BASE-T module, although it uses USB 3.0 PHY ICs which have both ULPI and PIPE3 interfaces joined via the FPGA.

Unlike our originally planned microcontroller-based solution, the Daisho USB 3.0 module supports SuperSpeed USB, the newest and fastest version of the USB protocol.

3.2.4 HDMI

HDMI is the only target for which standard PHY parts were unable to meet our needs. Most HDMI devices are highly integrated, partly due to the licensing and encryption protocols that IC manufacturers must adhere to, leaving few PHYs that provide an interface suitable for Daisho. These components support data rates more limited than we hoped to support with Daisho.

Some existing devices, such as the NeTV[6], directly interface the FPGA to the differential data lines of the HDMI signal. While possible, this requires an FPGA of sufficient power to process extremely high-speed (multiple gigabits per second) data, which introduces constraints on the video resolution.

Thankfully, HDMI data can be deserialized using widely available parts such that the data passed to the FPGA is a parallel representation of the physical layer data. While the total data rate remains unchanged, it is transmitted over many lower-speed parallel data lines. With this mechanism we expect to support video rates including 1080p and hope to support the new 4K standard as well.

In addition to video, HDMI can carry 100 Mbit/s Ethernet as well as low-speed data channels used for device configuration such as resolution negotiation. Any of these communication channels may be of interest for communication security research.

The HDMI front-end module utilizes two HDMI ports with full electrical protection, standard signal equalizers and amplifiers, and a general-purpose serializer/deserializer (SerDes). This SerDes is designed for extremely high data rate applications, such as 10 gigabit Ethernet, but features generic modes which we

are able to take advantage of. Although many FPGAs include high speed SerDes peripherals, we selected a separate SerDes IC in order to reduce mainboard cost.

Similar to the 1000BASE-T and USB3.0 modules, the FPGA copies the data between the HDMI ports, while forwarding a copy to the host. This minimizes latency and jitter introduced by the system.

3.3 Software

In keeping with the open source goals of Project Daisho, we try to ensure that all tools and software used in designing and operating Daisho are open source where possible or otherwise free of charge to the end user.

For hardware design we use KiCad[7] as it is the most advanced open source Electronic Design Automation (EDA) software available and members of the team have experience using it for previous projects.

Unfortunately there are no open tools for compiling Altera bitstreams, but the FPGA used for the Daisho boards can be used with the freely available, but not open source, Quartus II Web Edition[8] tools provided by Altera.

The host software, libdaisho, contains minimal control and data manipulation functions built on top of libusb 1.0. One of our continuing goals for libdaisho is to avoid any front-end specific functionality and simply allow tools on the host system to retrieve data from whichever front-end is attached. This pushes all target specific code in to the tools, leaving libdaisho lightweight and generic for any future front-end targets that are built.

Once tapped data is transferred to the host system, we use existing tools and standards such as Wireshark and the pcap file format to ensure consistency with other monitoring platforms. Wireshark support is made possible by the work that Kershaw and Ryan[9] have put in to extcap, an extension to the Wireshark codebase that allows the tool to pull data from any source in pcap format. This benefits not only Daisho but a host of other tools and projects.

Protocols such as RS-232 that are not packet based are not well suited to integration with Wireshark. For RS-232 we have written command line tools for analysis that can be extended by a researcher working with the technology.

To facilitate the collaborative and open process of designing and building the Daisho platform, we use GitHub[10] to host all of our designs and source code. We believe that this enables interested parties to view progress and contribute to the project.

4 Future Expansion

One of the biggest advantages to building a modular and extensible open source system is the freedom that it gives users to build their own front-end modules. There are many technologies that could benefit from low cost monitoring tools. These include DisplayPort, SAS, SATA, DVI, DSL and even as yet unreleased technologies.

More general monitoring devices could be built, for example a single front-end module that supports a variety of protocols that are often used together. A module that supports SPI, I2C, CAN and JTAG would provide an excellent general-purpose debugging tool.

The high bandwidth USB 3.0 link to a host system combined with the processing capabilities of the FPGA lend themselves well to wideband Software Defined Radio applications. It would be feasible to build a front-end module with large enough bandwidth to cover the entire 2.4GHz ISM band, capturing all Bluetooth or 802.11g data within range.

Further to the networking and security applications of Daisho, there are many possible uses of a base platform on which to build high bandwidth devices. Applications such as new communication protocols or experimental high resolution data capture techniques that need to move large volumes of data to or from a host system can benefit from being built upon Daisho.

5 Acknowledgments

This research would not be possible without the funding made available by the DARPA Cyber Fast Track program. We would like to thank our co-developers, Jared Boone, Marshall Hecht, and Benjamin Vernoux.

References

[1] http://greatscottgadgets.com/daisho/

[2] http://code.google.com/p/zigbee-security/

[3] http://greatscottgadgets.com/throwingstar/

[4] https://www.kernel.org/doc/Documentation/usb/usbmon.txt

[5] http://learn.adafruit.com/hacking-the-kinect/usb-analyzer

[6] http://kosagi.com/w/index.php?title=NeTV_Main_Page

[7] http://www.kicad-pcb.org

[8] http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html

[9] Expanding Wireshark Beyond Ethernet & Network Interfaces, Mike Kershaw and Mike Ryan, Sharkfest '13

[10] https://github.com/mossmann/daisho