Looping Surveillance Cameras Like in the Movies

Live modification of network streams without interruption of service

Zach Banks & Eric Van Albert

May 4, 2015

Abstract

This project consists of the hardware and software necessary to hijack wired network communications. The hardware allows an attacker to splice into live network cabling without ever breaking the physical connection. This allows the traffic on the line to be passively tapped and examined. Once the attacker has gained enough knowledge about the data being sent, the device switches to an active tap topology, where data in both directions can be modified on the fly. Through our custom implementation of the network stack, we can accurately mimic the two devices across almost all OSI layers.

We have developed several applications for this technology. Most notable is the editing of live video streams to produce a "camera loop," that is, hijacking the feed from an Ethernet surveillance camera so that the same footage repeats over and over again. More advanced video transformations can be applied if necessary. This attack can be executed and activated with practically no interruption in service, and when deactivated, is completely transparent.

1 Motivation

This project was undertaken because it appears quite frequently in the movies, but is rarely depicted in a remotely reasonable manner. We set out to re-create it as true to the movies as possible to demonstrate exactly how practical it would be to create a camera loop.

Figure 1: Camera loop devices from *Oceans Eleven* (2001, left) and *National Treasure* (2004, right).

2 Literature Review

There has been a fair amount of previous work done on the topic of surveillance camera hacking.[1][2] Because our attack requires physical access to the camera wire, it may seem less powerful than some existing attacks. However, accessing that wire may be more practical than adding a machine of your own to the network containing that camera.

We feel that our work is different in three main ways. First, it is more generic. It is not specific to a brand or model of camera. Because it works at the physical Ethernet level, it affects all Ethernet devices.

Second, our attack is more covert. We provide a hardware solution to gain access to the camera feed. This device is undetectable by software and may even be undetectable by network cable analyzers. Notably, it does not rely on any form of ARP spoofing or poisoning. The other attacks we have found involve making use of existing network access, which may show up in logs. These attacks can typically be defeated by simply keeping the surveillance cameras off of accessible networks or behind a firewall. Our attack works regardless of the network configuration and does not generate any ancillary network activity.

Finally, our attack is more extensible than any existing attack. With the universal tap board, any physical layer that uses 100Ω differential pairs can be tapped, as can any

[1] Black Hat 2013: *Exploiting Network Surveillance Cameras Like a Hollywood Hacker* — `https://www.youtube.com/watch?v=B8DjTcANBx0`

[2] DEFCON 17: *Advancing Video Application Attacks with Video Interception, Recording, and Replay* — `https://www.youtube.com/watch?v=QcsQ6UzMJiU`

low-speed signal. Likewise, our `lens` network framework allows nearly transparent editing of many common network communication protocols such as HTTP, and RTP, and can be expanded to include many more.

3 Hardware

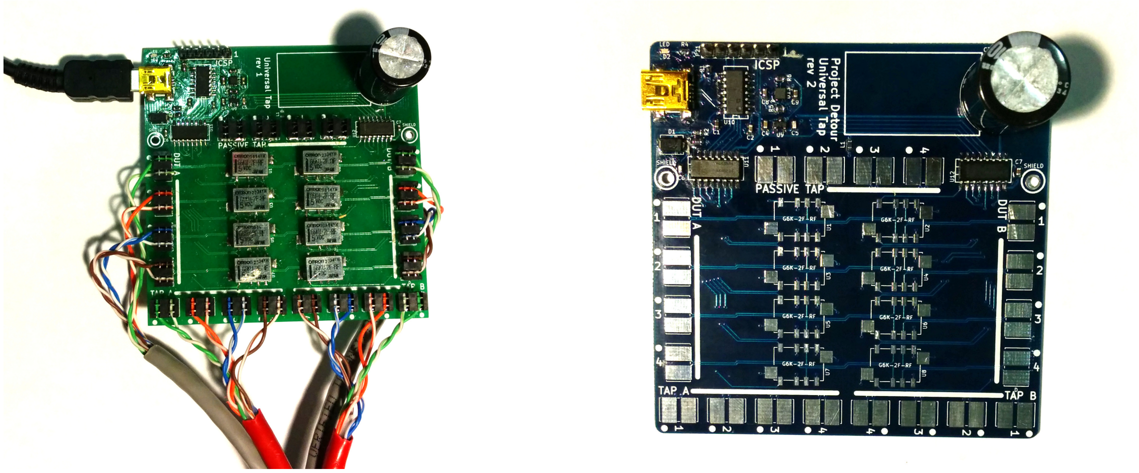
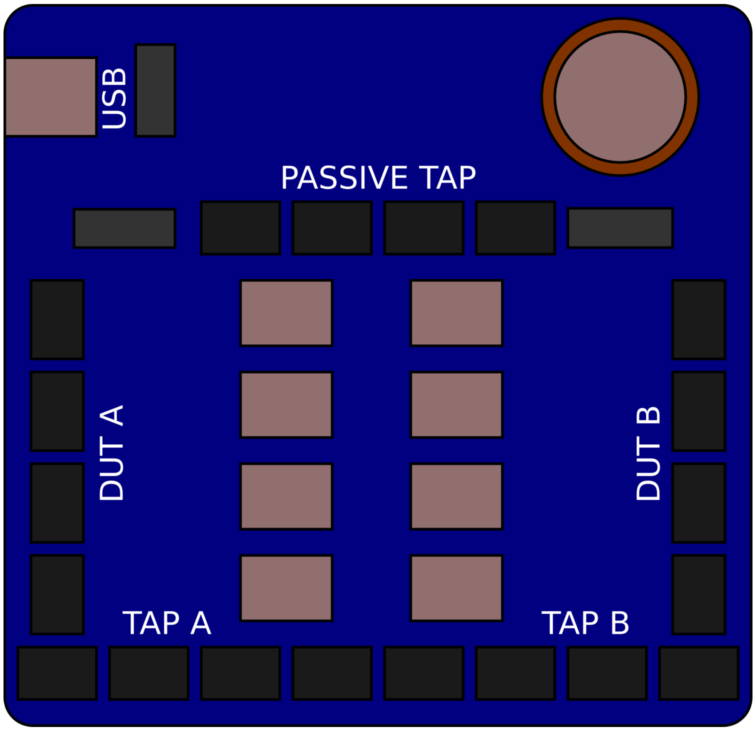
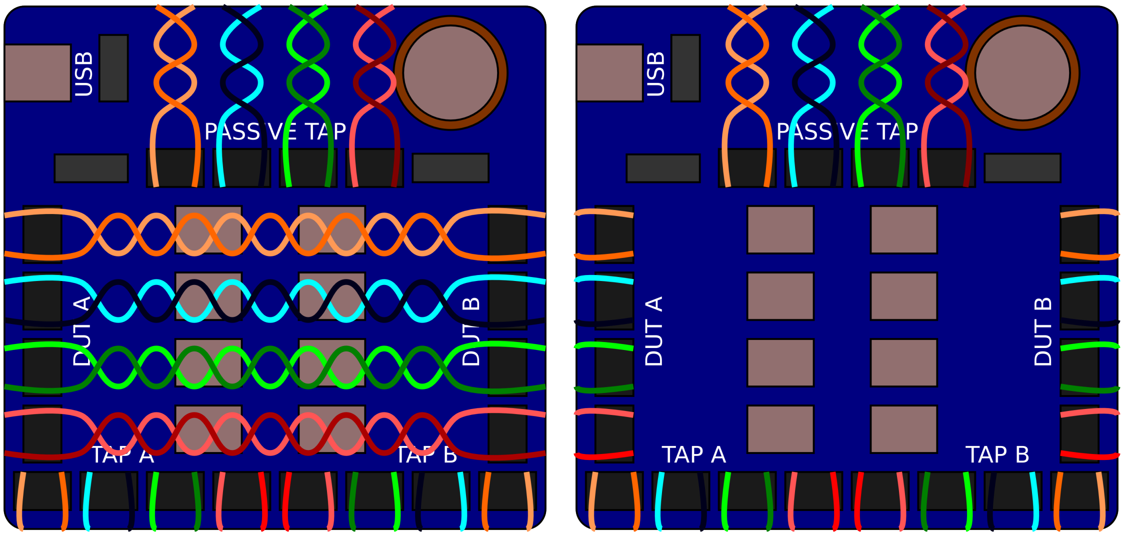


Figure 2: Two hardware revisions. On the left, an older version with cables spliced in. On the right, a new version in the process of being assembled



The hardware consists of a custom PCB which acts as a USB-controlled universal tap. It has five ports, each of which is four differential pairs. The ports are designated `DUT A`, `DUT B`, `TAP A`, `TAP B`, and `PASSIVE TAP`. A set of high-bandwidth relays control the routing of these ports. By default, `DUT A` (*Device Under Test A*) passes directly through to `DUT B` and tees off to `PASSIVE TAP`. Each port has punch-down connectors for wire insertion.

To insert the tap into a live cable, the line under test is spliced into both the `DUT A` and `DUT B` connectors. This adds a redundant copper path between the two devices. Passive monitoring is available on the `PASSIVE TAP` port. Once the tap has been connected, the existing cabling can be cut so that all signals are now routed through just the tap board.



The attacker then connects their man-in-the-middle device to the `TAP A` and `TAP B` ports. When commanded, the tap flips the relays to route `DUT A` to `TAP A`, and `DUT B` to `TAP B`. Traffic is now routed through the attacker's device in an active-tap configuration. The attacker is free to continue forwarding traffic transparently, or to generate fake traffic in either direction.

The tap board can be programmed to be fail-safe in the event of loss of USB communication or power. If a heartbeat signal from the host to device is lost, or power is disconnected, it can flip the relays back to transparent pass-through mode.

Finally, the tap board has a built-in accelerometer. It can use this in conjunction with a heartbeat signal from the device to the host to indicate tampering. If the board is unplugged or jostled from its resting position, the host will be alerted. The attacker can rest assured that the tap is in place, working, and undisturbed.

4 Software

Our software `lens` (*Live Editing of Network Streams*) aims to be as transparent and indetectable as possible while still enabling the attacker to capture, filter, modify, or spoof any packet. Although capturing packets is trivial, modifying or forging packets becomes increasingly difficult in higher OSI layers. To accomplish this, `lens` uses a custom network stack which allows the attacker to construct packets that are as similar as possible to those sent by the target devices. Additionally, `lens` is able to modify packets of stateful protocols (such as TCP) without interrupting the existing connection.

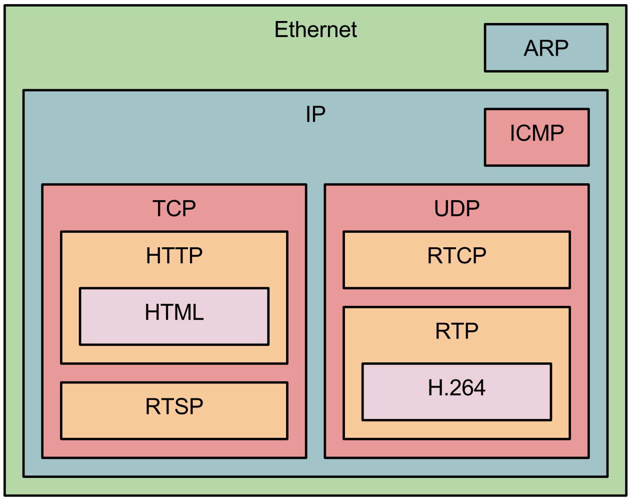


Figure 3: The beefy 5-layer OSI burrito

`lens` is implemented in Python (2.7) and is structured around tornados coroutine model. Each layer in the network stack is implemented separately and the layers are chained together in a pipeline-like fashion. Similar to the naitive network stack implemented by the operating system, each layer provides an abstraction to the next. For example, the TCP layer black-boxes the concept of open connections and their associated state for higher-up layers.

Starting with Ethernet, each layer decodes all packets that it recognizes. It examines the decoded data, and decides what to do. If it recognizes the protocol, it can pass it up the hierarchy for further decoding. If it doesnt recognize the protocol, it does nothing. Most layers unwrap one protocol layer, but they could just also apply transformations to the data. Ultimately, a layer will re-encode the data back to its original protocol and pass it back down the hierarchy.

Filter layers allow the attacker to perform transformations on given packets. For example, the infamous Cloud-to-Butt[3] extension can be mimicked with a `lens` filter which performs a simple find-and-replace on the text at the HTTP layer.

To perform the camera loop, we use a more advanced set of filters. The camera sends its network stream over RTSP (Real Time Streaming Protocol), which uses TCP to set up the video stream, RTCP (RTP Control Protocol) over UDP for synchronization, and RTP (Real Time Protocol) over UDP to transport the H.264-encoded video. Each layer in `lens` parses apart incoming packets, applies any modifications, and reconstructs outgoing packets.

[3] `https://github.com/panicsteve/cloud-to-butt`

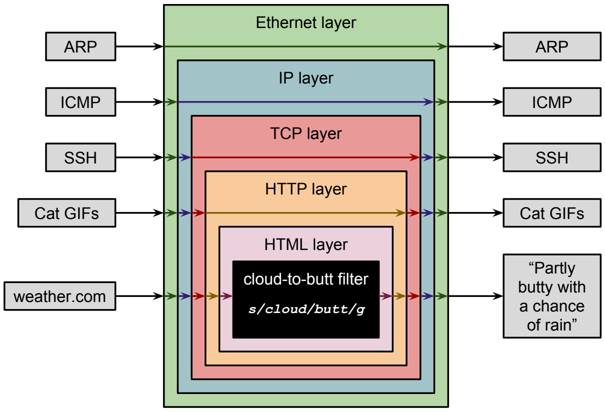


Figure 4: Graphical representation of `lens` mimicking the cloud-to-butt extension. Each arrow into a boundary indicates a software decoding or encoding step. A single simplex model is pictured here for clarity, however, `lens` can handle concurrent, duplex, and stateful transmissions.

In practice, video encoded with H.264 can be looped without re-encoding by concatenating the looped segment indefinitely. However, by re-encoding the stream with ffmpeg, an attacker can apply standard ffmpeg transformations to the video. For example, the attacker could loop most of the video, without modifying the corner of the frame that contains the timestamp.

5 Further Work

Using the `lens` framework and the universal tap board, an attacker can perform a camera loop attack on a networked camera without interrupting service. However, these tools can also be used to perform many other man-in-the-middle network attacks: such as TLS downgrade attacks or denial of service by packet filtering.

The universal tap board can also be used to tap cables other than Ethernet, such as HDMI. HDMI primarily sends data over four pairs of twisted differential pairs (1 clock, 3 data) which could be actively tapped with the universal tap board and modified with additional hardware. The tap board also supports simpler physical layers, such as USB (1 twisted pair) or Wiegand (2 single-ended lines).