Web-based Attacks on Local IoT Devices

See demo & paper:

https://goo.gl/2kxwe8

Gunes Acar **Danny Y. Huang** Frank Li* Arvind Narayanan Nick Feamster

Princeton University *University of California, Berkeley

Goals

Circumvent browser's single-origin policy

Discover certain IoT devices with JavaScript

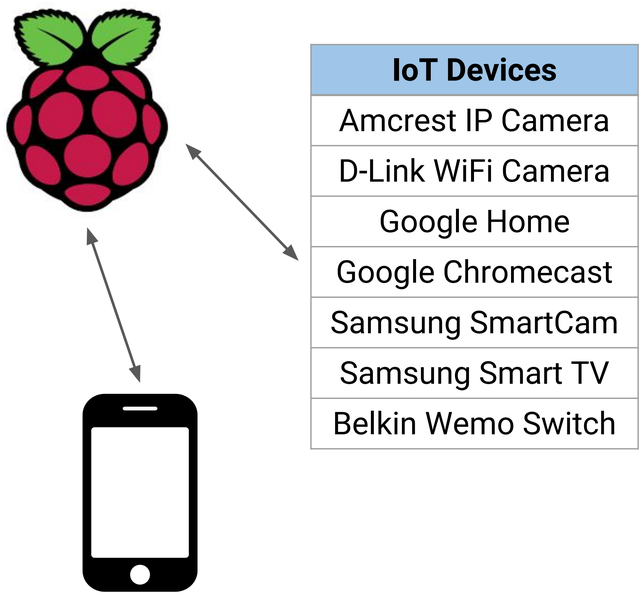
Access & control certain IoT devices with JavaScript

Preparing Attack

Set up a Raspberry Pi as a WiFi access point, connecting 15 IoT devices and an Android phone.

Interact with devices, taking pcaps at the RPi. Observed HTTP endpoints on 7 devices.

Searched for further documentation on HTTP APIs. Total: 35 GET, 8 POST.



IoT Devices
Amcrest IP Camera
D-Link WiFi Camera
Google Home
Google Chromecast
Samsung SmartCam
Samsung Smart TV
Belkin Wemo Switch

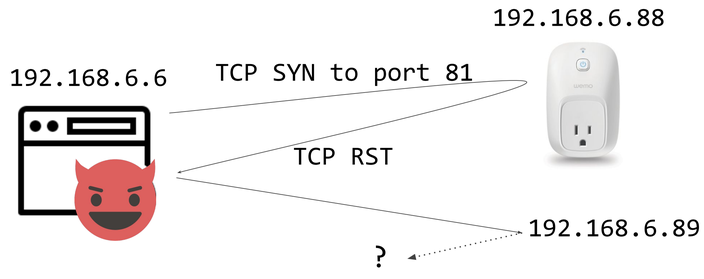
Attack 1: Discover Devices

Step 1: Find active local devices

Scan local subnet on port 81.

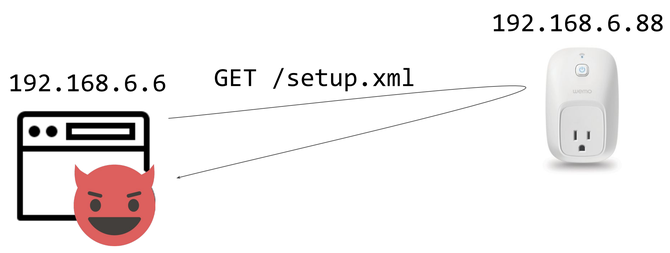
Use Fetch API to send GET.

Measure response time (TCP RST vs timeout).



Step 2: Identify IoT devices

a. Send request for our GET endpoints to active IP addresses, using HTML5 <audio> element.
b. Use resulting MediaError message to infer resource availability (new side channel).

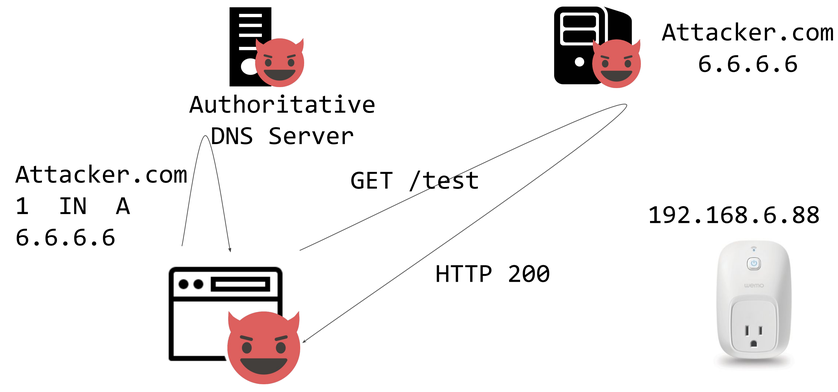


If Exists: `MEDIA_ERR_SRC_NOT_SUPPORTED` "`DEMUXER_ERROR_COULD_NOT_OPEN: FFmpegDemuxer: open context failed`"
Else: `MEDIA_ELEMENT_ERROR` "`Format error`"

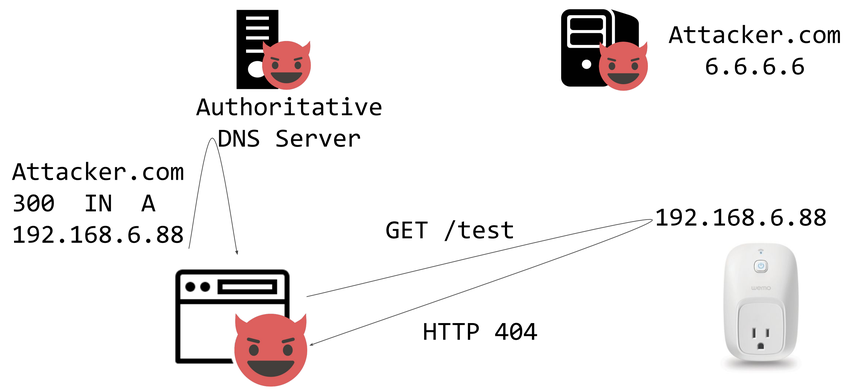
If Exists: `MEDIA_ERR_SRC_NOT_SUPPORTED` "`Failed to init decoder`"
Else: `MEDIA_ELEMENT_ERROR` "`Message 404: Not Found`"

Attack 2: Control Devices

Step 1: Victim visits attacker.com, queries malicious nameserver for attacker.com. Return web server IP with short TTL.



Step 2: Repeatedly visit attacker resource until cache expires.



Step 3: Attacker can directly access resources on targeted IoT device.

Summary

Privacy and security implications.

Problems can be mitigated by IoT vendors, DNS providers, ISPs, and browser vendors.

CENTER FOR INFORMATION TECHNOLOGY POLICY
AT PRINCETON UNIVERSITY