Web-based Attacks on Local IoT Devices

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


wemo


D-Link


SAMSUNG

FORTUNE

TECH • SAMSUNG

Samsung's smart fridge could be used to steal your Gmail login

Forbes

Billionaires Innovation Leadership Money Consumer Industry

A Massive Number Of IoT Cameras Are Hackable -- And Now The Next Web Crisis Looms

Thomas Fox-Brewster Forbes Staff
Security

Samsung and Roku Smart TVs Vulnerable to Hacking, Consumer Reports Finds

Security and privacy testing of several brands also reveals broad-based data collection. How to limit your exposure.

By Consumer Reports
February 07, 2018

2.8K SHARES

Consumer Reports has found that millions of smart TVs can be controlled by hackers exploiting easy-to-find security flaws.

The problems affect Samsung televisions, along with models made by TCL and other brands that use the Roku TV smart-TV platform, as well as streaming devices such as the Roku Ultra.

Call to ban sale of IoT toys with proven security flaws

Natasha Lomas @riptari / Nov 15, 2017

Comment

How to reach local IoT devices?

Public devices (e.g., port forwarding)

Local malware

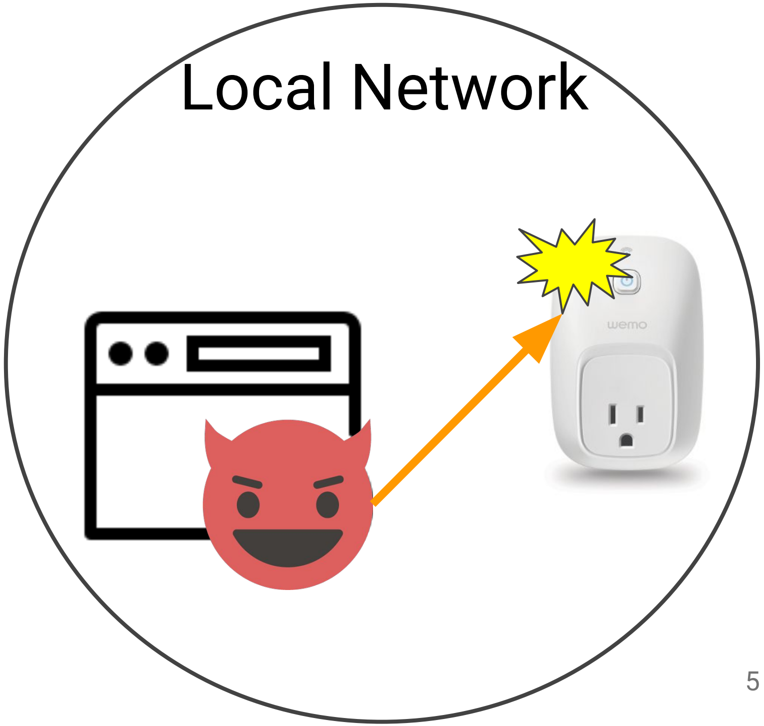
Web attacks (*this paper*)

How to reach local IoT devices?

Public devices (e.g., port forwarding)

Local malware

Web attacks (*this paper*)



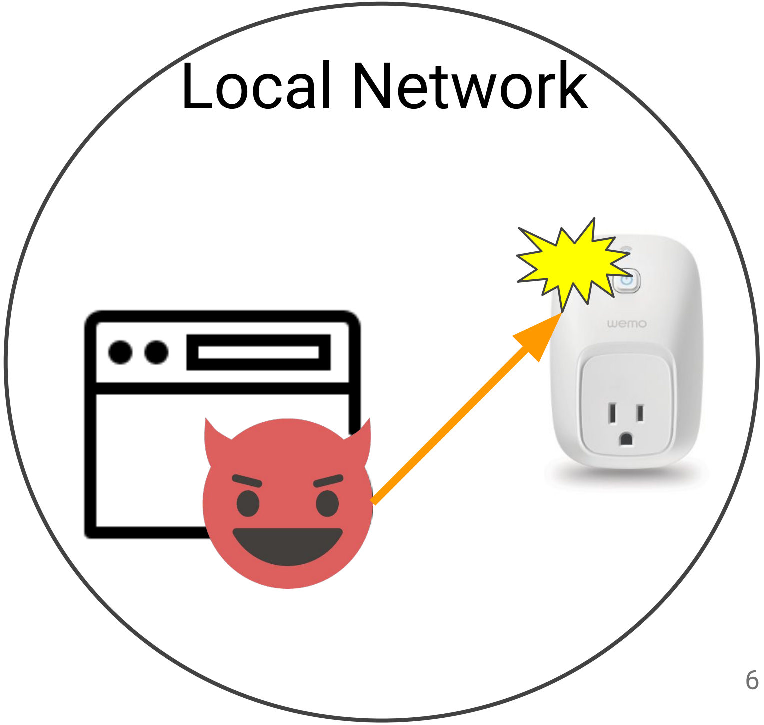
How to reach local IoT devices?

Public devices (e.g., port forwarding)

Local malware

Web attacks (*this paper*)

1. Discover certain IoT devices
2. Access & control IoT devices



Preparing the Attacks

Targeting HTTP Servers

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

Targeting HTTP Servers

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

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

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

Targeting HTTP Servers

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

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

3. Searched for further documentation on HTTP APIs
   a. 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

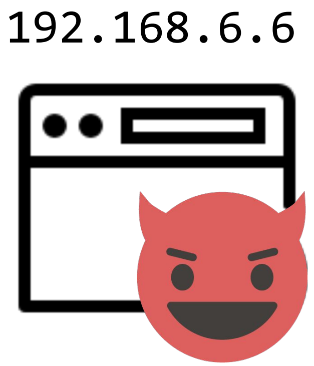
Attack 1:

Identify Local IoT Devices

Attack Steps

Attack Steps

1. Get local IP (via WebRTC SDP)



Attack Steps

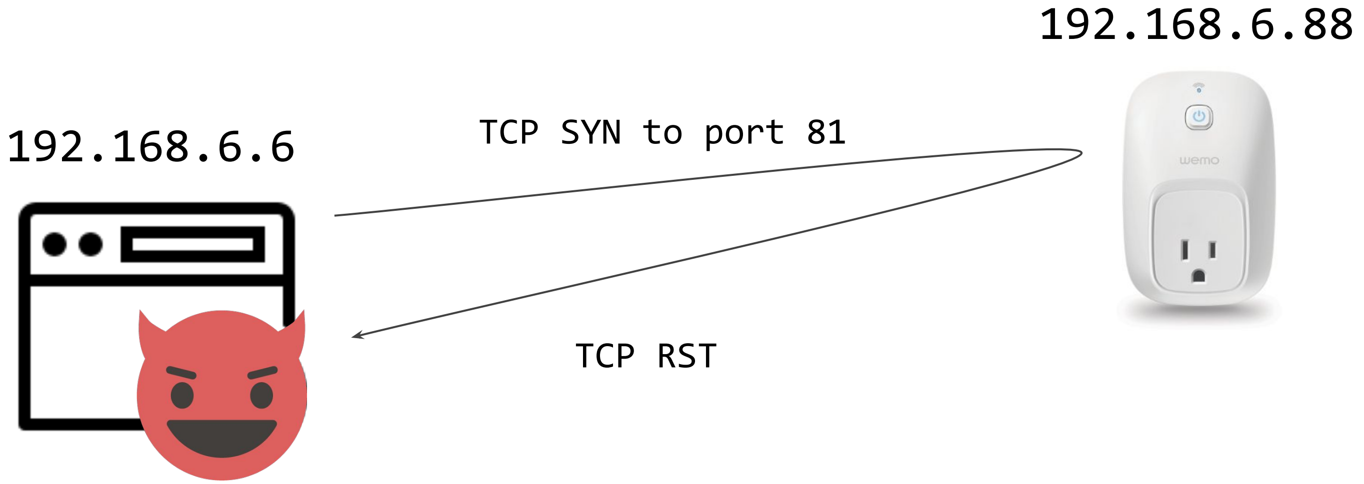
2. Find active local devices.
    a. Scan local subnet on port 81, sending GET request (via Fetch API)
    b. Measure response times (TCP RST vs TCP timeout)

192.168.6.88

192.168.6.6

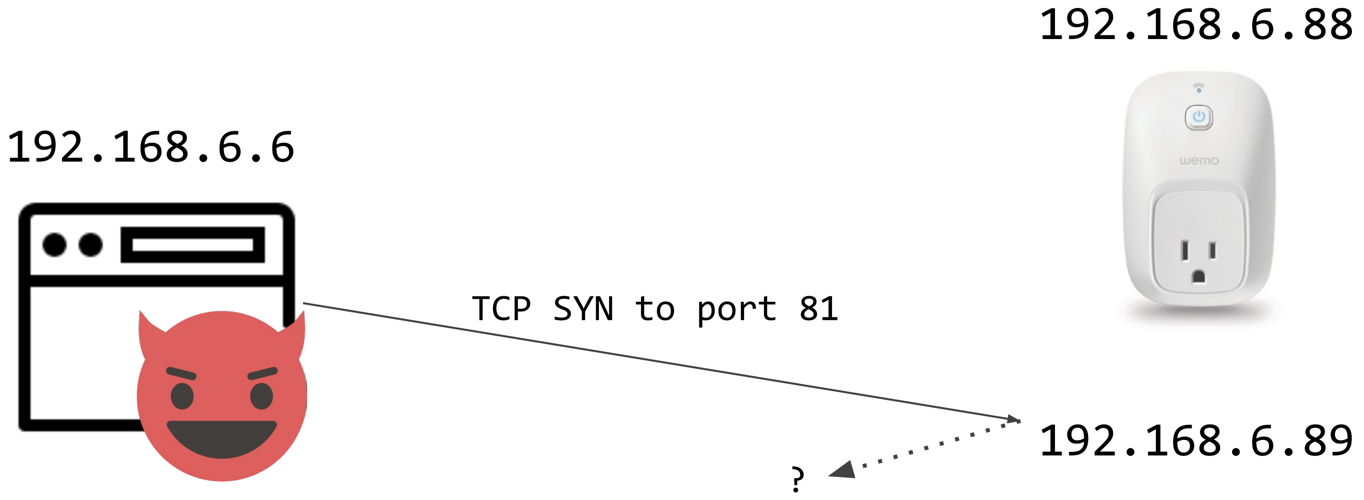
Attack Steps

2. Find active local devices.
    a. Scan local subnet on port 81, sending GET request (via Fetch API)
    b. Measure response times (TCP RST vs TCP timeout)



Attack Steps

2. Find active local devices.
    a. Scan local subnet on port 81, sending GET request (via Fetch API)
    b. Measure response times (TCP RST vs TCP timeout)



Attack Steps

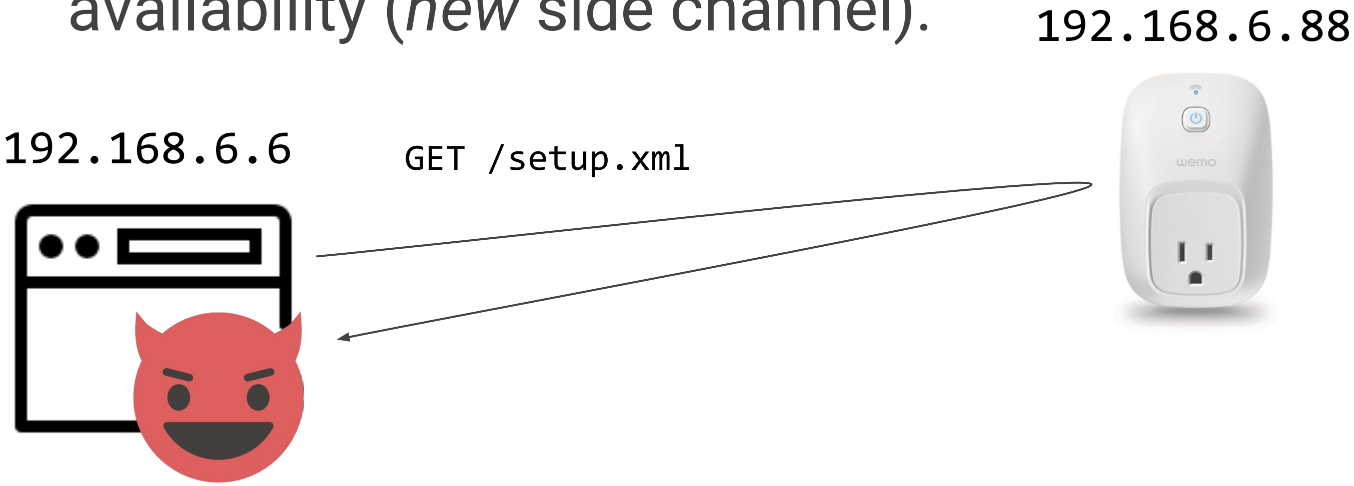
3. Identify IoT devices.
   a. Send request for our GET endpoints to active IP addresses, using HTML5 <audio> element.
   b. Use resulting MediaError message to infer resource availability (*new* side channel).

192.168.6.88

192.168.6.6

Attack Steps

3. 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).



Attack Steps

3. 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”`

Attack Steps

3. 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 “Failed to init decoder”`

Else: `MEDIA_ELEMENT_ERROR “Message 404: Not Found”`

Attack Steps

3. 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).

Safari: Fetches timed out
Edge: No MediaError error messages
(Attack 1 does not work)

Implications

Side-channel sidestepping SOP (Chrome bug bounty)

Attack stepping stone

Privacy leaks (e.g., network fingerprinting)

Attack 2:

Access & Control Local Devices

DNS Rebinding

Attack fully bypassing SOP

(D. Dean, E. Felten, and D. Wallach, IEEE S&P 1996)

Requires a web attacker (controls malicious domain + webserver) <u>also</u> controlling domain's authoritative DNS nameserver

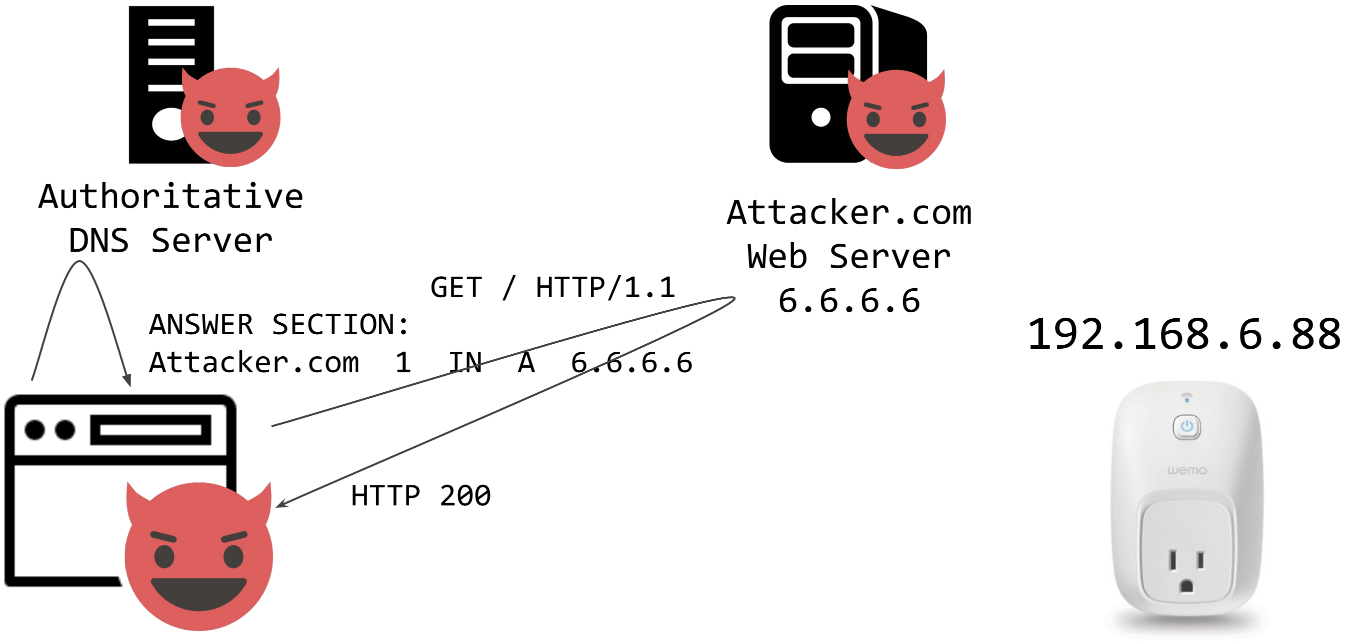
Attack Steps

Attack Steps

192.168.6.88

Attack Steps

1. Victim visits *attacker.com*, queries malicious nameserver for *attacker.com*. Return web server IP w/ short TTL.

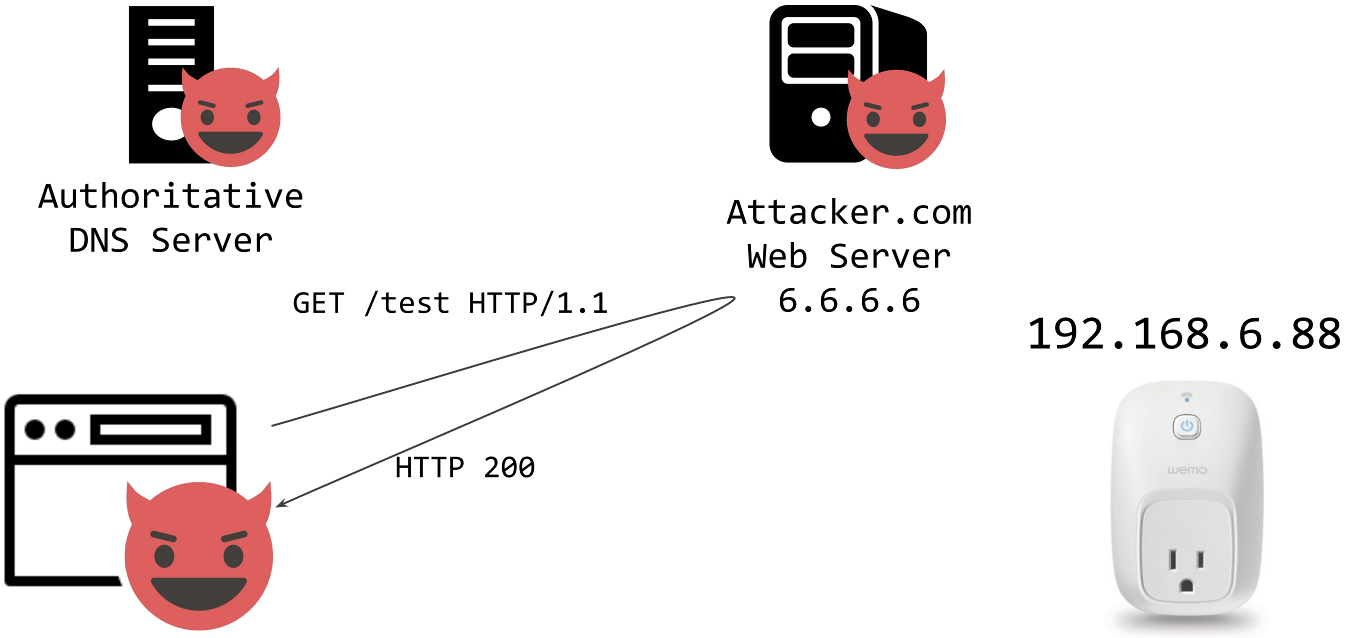


Attack Steps

2. Attacker website loads another resource *test*.

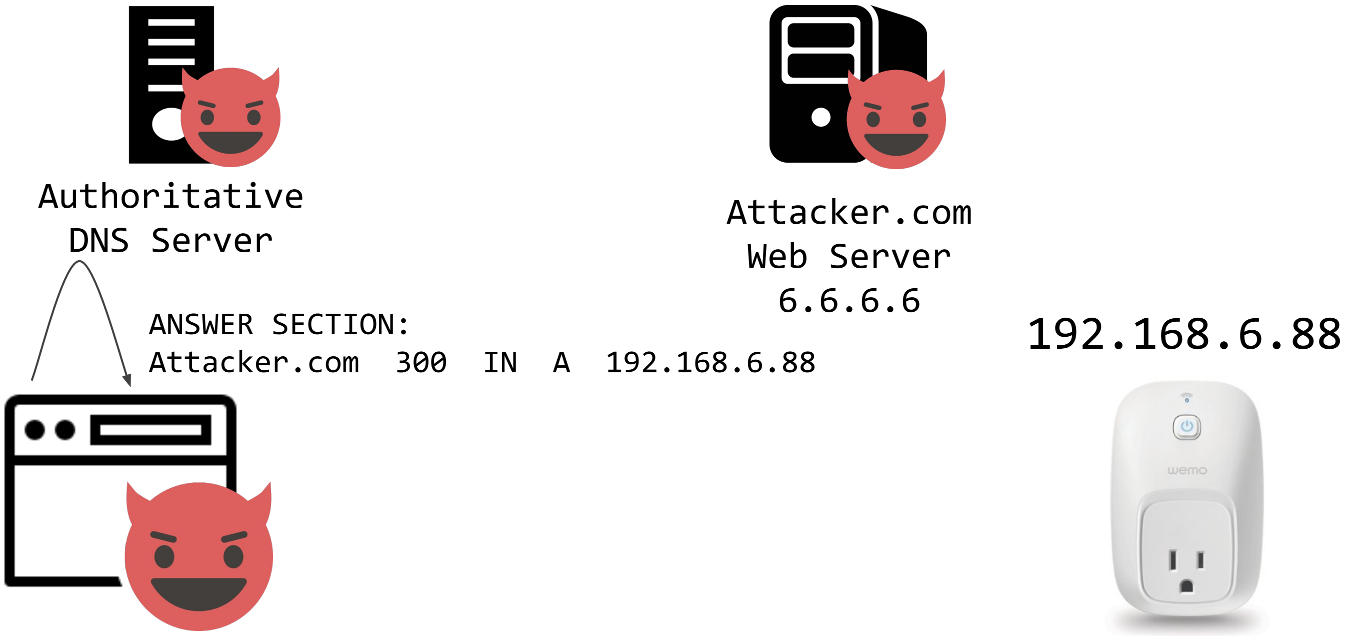
Attack Steps

3. If *attacker.com*'s DNS record is cached, *test* is directly retrieved. If so, wait and retry...



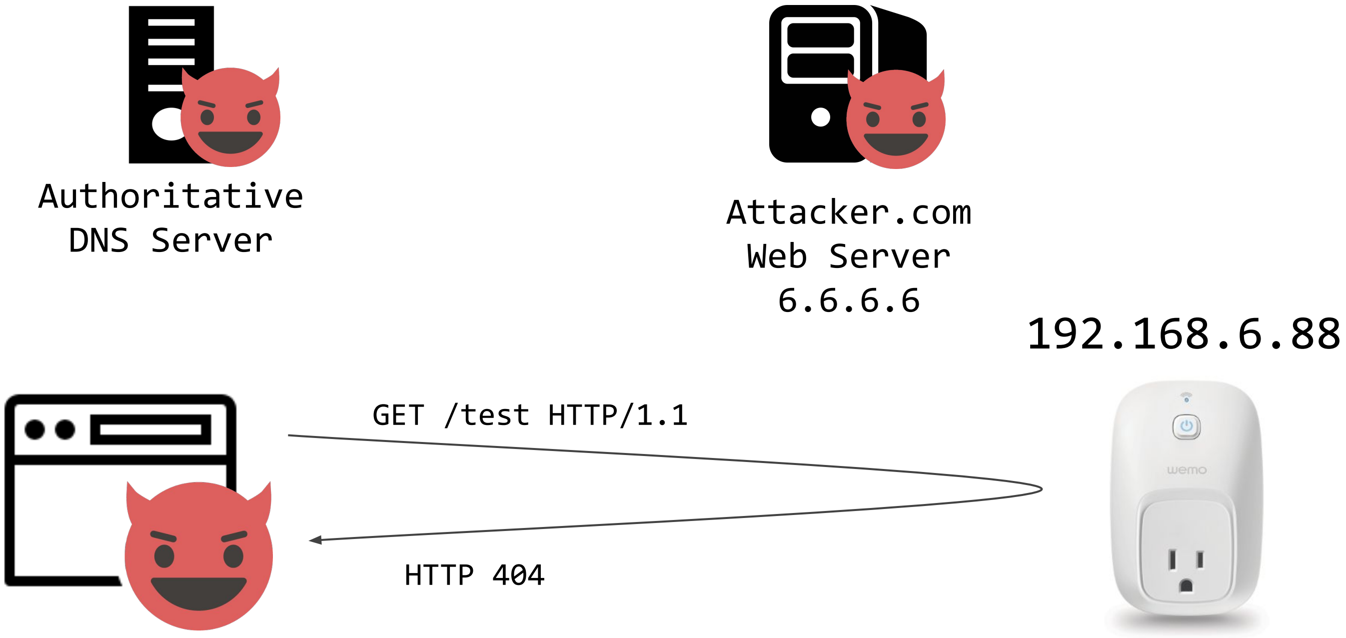
Attack Steps

4. If *attacker.com*'s DNS record is *not* cached, browser queries malicious nameserver again. Now return target IP w/ large TTL.



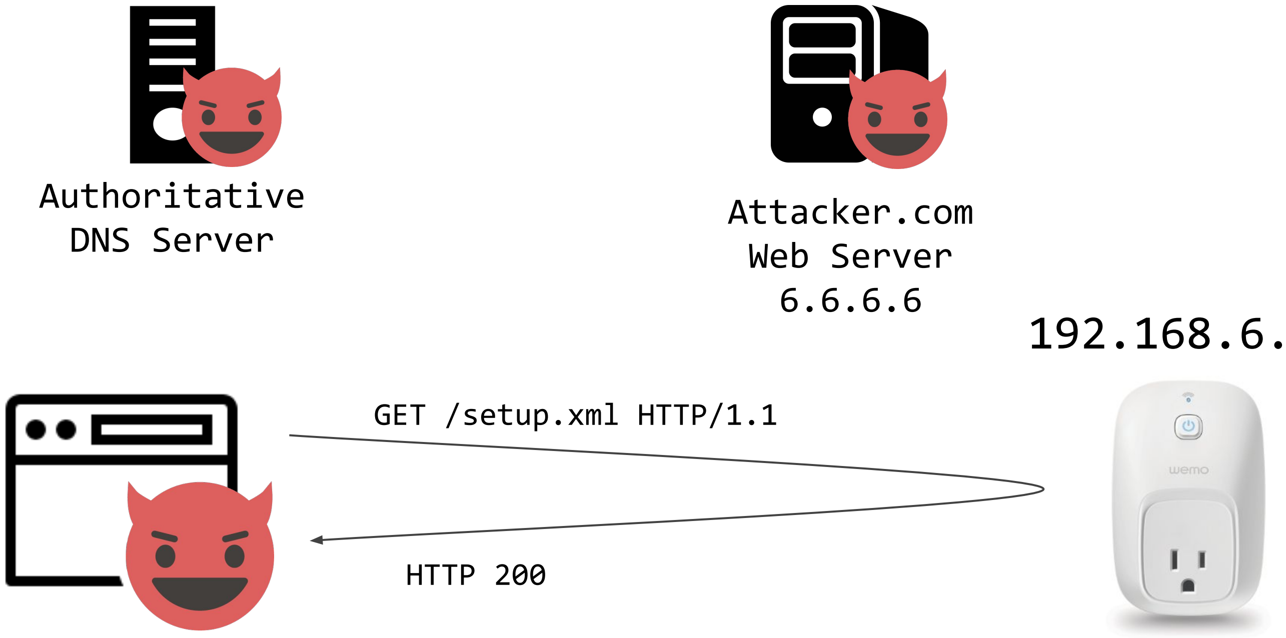
Attack Steps

5. This time, retrieving *test* fails. But *attacker.com* is now rebound to the target IP, and can make direct requests.



Attack Steps

5. This time, retrieving *test* fails. But *attacker.com* is now <u>rebound</u> to the target IP, and can make direct requests.



Attack on Devices

Attack on Devices

Google Home/Chromecast

Potential attacks:

- Play arbitrary Youtube videos on Chromecast
- Reboot Chromecast/Home
- Scan for WiFi networks and return information

Attack Demo

Implications

Attacker control of IoT device actions

Exploiting IoT device vulnerabilities for full compromise

Privacy leaks (e.g., extensive device fingerprinting or user profiling)

Moving Forward...

- Low barrier to attacks on local IoT devices via malicious websites.
- Need defenses that protect against lateral attacks.

Thank you

https://iot-inspector.princeton.edu/

frankli@cs.berkeley.edu
@frankli714

Attack 1 Countermeasures

Home Users:

- Disable getting local IP via WebRTC SDP
- Configure DHCP to allocate for a larger subnet (e.g., /16)

Browsers:

- Limit private IP access for web pages with public domains

IoT Vendors:

- Respond to all GET request with 200 OK code

Attack on Devices

Attack on Devices

Google Home/Chromecast

Attack on Devices

Google Home/Chromecast

Access:

- Unique device ID
- Build/firmware version
- SSID of connected WiFi network
- Device schedules/alarms (Home)

Attack on Devices

Google Home/Chromecast

Control:

- Reboot device
- Play any video (Chromecast)
- Scan for WiFi networks and return SSIDs detected

Attack 2 Countermeasures

Home Users:

- Enable DNS forwarding with rebind protection

Browsers:

- Unclear?

IoT Vendors:

- Filter/validate based on HTTP headers

DNS providers:

- Filter private IPs from DNS responses

HTTP endpoints - examples

- **DlinkCamera -** *GET* `http://IP-ADDRESS:80/common/info.cgi`
- **Response**:

```
model=DCS-5020L
brand=D-Link
version=1.14
build=9
hw_version=A
name=DCS-5020L
location=
macaddr=B0:C5:54:0C:D2:74
ipaddr=172.24.1.99
netmask=255.255.255.0
gateway=172.24.1.1
wireless=yes
ptz=P,T
inputs=0
outputs=0
speaker=no
videoout=no
```

HTTP endpoints - examples

Get all WiFi networks on WeMo switch:

```
http://IP-ADDRESS:49154/upnp/control/WiFiSetup1 {"method": "POST", "body": "<?xml
version='1.0'?><SOAP-ENV:Envelope
xmlns:SOAP-ENV='http://schemas.xmlsoap.org/soap/envelope/'
SOAP-ENV:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'><SOAP-ENV:Body>
<m:GetNetworkList xmlns:m='urn:Belkin:service:WiFiSetup:1'>
</m:GetNetworkList></SOAP-ENV:Body></SOAP-ENV:Envelope>", "headers": {"Content-Type":
"text/xml", "SOAPAction": "\"urn:Belkin:service:WiFiSetup:1#GetNetworkList\""}}
```

Returns all nearby Wifi networks

HTTP endpoints - examples

- **Play arbitrary videos on Google Chromecast** - POST `http://IP-ADDRESS:8008/apps/YouTube {"method": "POST", "body": "v=oHg5SJYRHA0", "headers": {"User-Agent": "blah"}}`
- **Reboot Google Home and Chromecast** - `http://172.24.1.51:8008/setup/reboot {"method": "POST", "body": "{\"params\": \"now\"}", "headers": {"User-Agent": "blah", "Content-Type": "application/json"}}`

Results

IoT Device	*Attack*
Amcrest HD Series IP Security Camera	①
D-Link Wifi Camera	① ②
Google Home	① ②
Google Chromecast	① ②
Samsung SmartCam HD Pro	① ②
Samsung UHD Smart TV	① ②
Belkin Wemo Smart Switch	① ②

Table 1: IoT devices with open HTTP servers, and to which attacks (① and/or ②) they are vulnerable.

Attack 2

Capabilities	*C*	*D*	*H*	*S*	*T*	*W*
Get Software Version or Model	✓	✓	✓	✓	✓	✓
Get Current SSID	✓		✓	✓	✓	✓
Get Nearby SSIDs	✓		✓	✓		✓
Get Device Unique Identifier	✓	✓	✓	✓	✓	✓
Get Owner's Username				✓		
Change State	✓		✓		✓	✓

Table 3: What Attack ② could do to IoT devices: Google [C]hromecast, [D]-Link Camera, Google [H]ome, Samsung [S]martCam, Samsung [T]V, and [W]emo Switch.

Attack 2: Which OSes and browsers are vulnerable

OS	*Request*	*Chrome*	*Firefox*	*Safari*
Ubuntu	GET	C D H S T W	C $\underline{\text{D}}$ H $\underline{\text{S}}$ $\underline{\text{T}}$ $\underline{\text{W}}$	N/A
	POST	$\underline{\text{C}}$ $\underline{\text{H}}$ T W	C H $\underline{\text{T}}$ $\underline{\text{W}}$	N/A
macOS	GET	C D H S T W	C *D* H S *T* $\underline{\text{W}}$	C D H S T W
	POST	$\underline{\text{C}}$ $\underline{\text{H}}$ T W	C H T $\underline{\text{W}}$	$\underline{\text{C}}$ $\underline{\text{H}}$ T W
Windows	GET	C D H S T W	C D H S T W	N/A
	POST	$\underline{\text{C}}$ $\underline{\text{H}}$ T W	C H T W	N/A

Table 4: Which operating systems and browsers were vulnerable to Attack ② against the following devices: Google [C]hromecast, [D]-Link Camera, Google [H]ome, Samsung [S]martCam, Samsung [T]V, and [W]emo Switch. An unformatted letter indicates that the attack was successful on all known HTTP endpoints on a given device; an underline indicates unsuccessful attacks on all of the HTTP endpoints; and italics indicates that some of the endpoints were vulnerable to our attack. We omit listing Microsoft Edge as all attacks failed on it.

Responsible Disclosure

- We reported the vulnerabilities to...
  - Browser vendors: Chromium (Google), Mozilla
  - IoT vendors: Google, Samsung, D-Link, Belkin
- Chromium offered bug bounty of $500
  - Fixed, will be released in v68
- Mozilla bug is still "*Unassigned*"
- Google Home: known issue
- Belkin promised to release a patch in August
- Ack from Samsung
- No response from D-Link