Smart-Phones Attacking Smart-Homes*

Vijay Sivaraman
Univesity of New South Wales
Sydney, NSW, Australia
vijay@unsw.edu.au

Dominic Chan,Dylan Earl
Univesity of New South Wales
Sydney, NSW, Australia
dominiczchan@gmail.com,
dylan.earl@me.com

Roksana Boreli
National ICT Australia
Sydney, NSW, Australia
Roksana.Boreli@nicta.com.au

ABSTRACT

The explosion in Internet-connected household devices, such as light-bulbs, smoke-alarms, power-switches, and webcams, is creating new vectors for attacking "smart-homes" at an unprecedented scale. Common perception is that smart-home IoT devices are protected from Internet attacks by the perimeter security offered by home routers. In this paper we demonstrate how an attacker can infiltrate the home network via a doctored smart-phone app. Unbeknownst to the user, this app scouts for vulnerable IoT devices within the home, reports them to an external entity, and modifies the firewall to allow the external entity to directly attack the IoT device. The ability to infiltrate smart-homes via doctored smart-phone apps demonstrates that home routers are poor protection against Internet attacks and highlights the need for increased security for IoT devices.

1. INTRODUCTION

The Internet-of-Things (IoT) is growing at a rapid rate: Gartner predicts that deployments will grow from 5 billion in 2015 to 25 billion by 2020 [11]. The boom in Internet-connected household devices, such as light-bulbs, cameras, smoke-alarms, and door-locks, is fueling the growth of the "smart-home"; indeed surveys [14] indicate that 51% of people in the US are willing to pay in excess of $500 for a well-equipped smart-home, with family safety, property protection, lighting/energy management, and pet monitoring as top motivators. While the smart-home brings huge benefits to consumers, who can lock/unlock doors from miles away, get instant alerts when smoke is detected in the house, and control lighting systems remotely, it is accompanied by substantial risks to privacy and security: hackers have been known to intrude on the home via baby-monitor cameras [10], and even take control of light-bulbs [9] and power-switches [16] remotely.

*This work was funded by the Australian Research Council (ARC) grant DP150100564.

WiSec'16 , July 18-22, 2016, Darmstadt, Germany

DOI: http://dx.doi.org/10.1145/2939918.2939925

Manufacturers have unfortunately been lax in embedding appropriate security protections in their consumer IoT devices, due to multiple reasons: business pressures force them to rush to market, revenues are derived from unit-sales rather than ongoing service, and security measures require skills and resources that add to costs. In spite of poor security on IoT devices shipping today, there is fortunately security at the perimeter of the network where they are deployed - a typical broadband router/gateway used in the home today, by virtue of its in-built NAT and firewall capability, prevents outside entities from launching gratuitous attacks on IoT devices inside the home network. For example, IoT devices like the Phillips Hue light-bulb and Belkin WeMo power-switch can be controlled with little or no authentication credentials [19], but are saved from being attacked openly on the Internet today by virtue of the home gateway behind which they sit; this is a consequence of the fact that an incoming packet would bear the public IP address assigned to the house, and the gateway would not know which of the multiple devices in the house, each with its own private IP address, to send the packet to. This "firewall" feature, a side-effect of network address translation (NAT) between the public and private IP addresses, protects IoT devices in the home from direct Internet attacks.

We believe that the inherent perimeter security provided by the home gateway is breeding complacency about the vulnerability of the smart-home to Internet attacks. In this paper we argue that the NAT/firewall protection is somewhat illusory, and can be easily penetrated by malware on users' smart-phones. We take an existing application from Apple's AppStore, instrument it to include malware, and get it approved as a legitimate app. We intentionally chose the Apple platform since it has tighter restrictions on what an app can do, and a more stringent approval process, than the Android platform. We then install and operate the instrumented app within selected homes (no human subjects were used for this trial, other than the members of this project team), and show how we can trigger it to scout for IoT devices in the home and report them to a server we operate in the cloud. The "reconnaissance" performed by our malware, which could not have been done from outside the home network, gives the attacker information about the IoT landscape within the house. Armed with this information, we then show how specific devices within the home can be attacked from outside. Our malware, when triggered appropriately, communicates with the home gateway (using Universal Plug-n-Play or UPnP) to modify firewall settings so that Internet traffic directed to a specific port is forwarded

to the victim IoT device, thereby allowing arbitrarily crafted attacks to penetrate the home. Once done, the malware can restore firewall configuration to remove trace of the attack, or keep it open for future attacks.

We show that all the above are real, not hypothetical. Our instrumented app was on the Apple AppStore (for only a brief period due to ethical reasons), we used it to discover several IoT devices in multiple homes, we have used it to surreptitiously modify firewall configuration on home gateways from multiple vendors, and we have demonstrated how an attacker can compromise multiple IoT devices (including a Belkin WeMo power switch and a D-Link IP camera) previously thought secure behind NAT/firewall. Our attack method is general, in that it can be applied to a wide range of IoT devices, and can be evolved to exploit new vulnerabilities as they are discovered, without having to upgrade the app. Our demonstration of the "infiltration" of the smart-home via a smart-phone app raises the prospect that the security provided by home gateways may be illusory, and the threat to smart-home IoT devices from large-scale Internet attacks more real than thought before.

The rest of this paper is organized as follows: §2 reviews threats to smart-home IoT devices and current defense approaches. In §3 we outline the design and implementation of our attacks that bypass the home perimeter security, and demonstrate and evaluate its performance in §4. The paper is concluded in §5 with a discussion on the impact for emerging smart-homes.

2. BACKGROUND

2.1 IoT Security Threats

The vast heterogeneity in smart-home IoT devices makes their attack vectors large, and it very challenging to map out the entire threat space. Prior works have exposed serious security flaws in numerous smart-home devices: for example our earlier work [19] shows that Internet-connected smart-bulbs and power-switches are easily compromised because they have poor authentication controls, while [12] shows that digital photo-frames, cameras, and speakers transmit data in plain-text that is easy to snoop upon to compromise user privacy. In addition to the above security flaws that have been revealed in researchers' labs, there is growing evidence of large-scale real-world security breaches: in Jan 2014 it was reported that a smart-fridge was among 100,000 devices that were compromised to send out spam emails [17]. As the adoption of smart-homes increases, security of IoT becomes a growing concern.

2.2 IoT Security Defenses

The growing importance of IoT security has led to a flurry of activity to develop device-level solutions, both by large device manufacturers and by standards bodies: for example, security frameworks are being developed by the Online Trust Alliance [3], the M2I security framework [2], IEEE P2413 [1], and Google Brillo/Weave [5], to name but a few. While these are worthwhile efforts, they require the security solutions to be embedded in the IoT device, which will take a long time to mature and gain wide adoption. In the meantime, researchers are developing non-embedded solutions that can protect IoT devices by inspecting traffic at the network level [21, 22]. Such efforts are still in the early research stages and not ready for deployment.

The only thing that protects insecure devices like light-bulbs, power-switches, webcams, photo-frames, etc. in the smart-home today is the home router. As mentioned earlier, the home router, by virtue of its NAT functionality that translates between the public and private IP addresses, drops unsolicited traffic from the Internet entering the home. This not only prevents Internet attackers from accessing the device, but also hides them so an attacker does not even know what IoT devices are in the home. We will show in this paper that this over-reliance on the home router is dangerous; an attacker can infiltrate the smart-home using malware on the user's smart-phone, and once on the inside, can not only scout for vulnerable devices, but also expose them to external attack with ease.

3. ATTACK DESIGN & IMPLEMENTATION

The objective of our attack is to bypass the perimeter security in home routers. A home router typically translates between the single external-facing public IP address assigned to the house and multiple internal-facing IP addresses assigned to devices within the home. A side-effect of this network address translation (NAT) is that unsolicited traffic from the Internet cannot penetrate the house, thereby providing firewall perimeter security. Our approach to penetrating this perimeter security is to embed malware into smart-phone apps that the user unwittingly runs inside the home network.

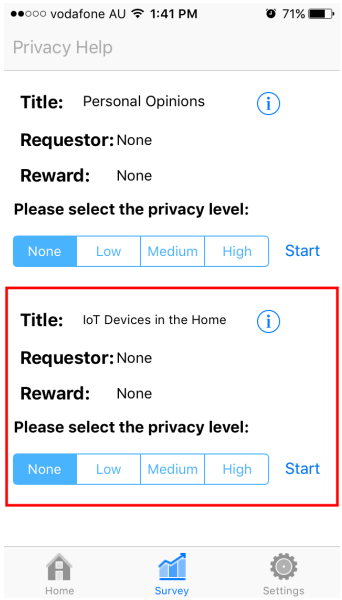


Figure 1: Survey app with malware trigger

3.1 The iPhone App

Mobile apps are susceptible to tampering, and App piracy is big business. We chose to work with iOS to demonstrate our attacks, since Apple operates a more secure ecosystem than Android; indeed independent analysis by MetaIntelli in 2015 found that over 90% of the 96,000 Android apps analyzed from the Google Play store had unprotected binary code [6], while Nokia's malware report for 2015 shows that 18 out of the 20 top smartphone infections were on the Android platform [18]. Our malware was inserted into an existing

privacy-preserving survey app in the Apple AppStore, called Loki, resubmitted as a version upgrade, and was approved by Apple for release. We briefly describe the App and how it meets our design objectives.

Front end: We used a preexisting app to conceal our malware. This app has been on the AppStore for two years and was designed for users to take surveys. We chose this app as it had: (a) previously been released, so we could be sure that the underlying app itself would not affect the result of the app review process of our malware laden version, (b) legitimate uses of the networking APIs to fetch surveys, so that network activity by the malware would not arouse suspicion, and (c) no active user base so it was extremely unlikely that someone outside our group would download the malware laden app in the short period it was available on the AppStore. Our use of a preexisting app also demonstrates the ease with which our malware is embedded. As there is no tight integration between the user facing app and our malware it is feasible that our malware could be attached to apps en-masse, as has occurred with the case of the XcodeGhost malware [8].

Trigger: Packet sniffing of our modified survey app would reveal that our malware was scouting for devices, a functionality that is not typical in a survey app. In order to minimize the chance of detection by Apple in the app review process, we suppressed the malware from starting until a trigger condition had been met. Because our malware is embedded in a survey app we decided that an appropriate trigger condition would be the selection of a survey with the trigger phrase "IoT Devices", as illustrated in Fig. 1. By manipulating the list of surveys available at the server which delivers the surveys to the app, we are able to remote-control the triggering of the malware. Our use of the trigger phrase above is meant to target users who have some knowledge of the Internet of Things and hence more likely to own IoT devices, but any arbitrary trigger can be used in general.

Secrecy: Our malware performs network scans and transmissions, which can interfere with the responsiveness of the app, affecting user experience. We originally implemented the device discovery code in a synchronous manner, using POSIX sockets that blocked the user interface. Our initial experiments revealed that this did not scale well to a large number of IoT devices, and led to noticeable degradation in user experience. We therefore reimplemented our IoT device discovery process using the asynchronous `CocoaAsyncSocket` library, a socket wrapper that runs in a separate thread. This not only allows our app to scale to a large environment with many IoT devices, but also better decouples the front-end app from the malware, allowing the malware to be embedded more easily into other apps.

Generality: Our malware is designed to be a general tool that an attacker can use to target a multiplicity of IoT devices, not just one specific IoT device. This allows the attacker to target new IoT devices as they emerge in the market, or existing IoT devices with updated firmware, without requiring the user to upgrade the infected app. Our malware therefore has limited embedded intelligence; instead it works in conjunction with a cloud-hosted server that holds the attack logic for various IoT devices. This approach makes the attack many-fold more effective and scalable than a local attack on a specific IoT device from the app itself.

Functions: Our malware performs two functions. The first part, elaborated in §3.2, is able to scout the local network for IoT devices and relay this information back to our attack server. The scouting is done using Simple Service Discovery Protocol (SSDP), and information on discovered devices is uploaded to our server through an HTTP POST API. The second function, detailed in §3.3, is able to configure port mappings on the home router in order to give the external attack server direct access to a specific IoT device in the home. The malware fetches the appropriate instruction from our attack server using an HTTP GET request, and executes the port mapping on the home router by using a UPnP command that residential Internet gateway devices from most vendors support.

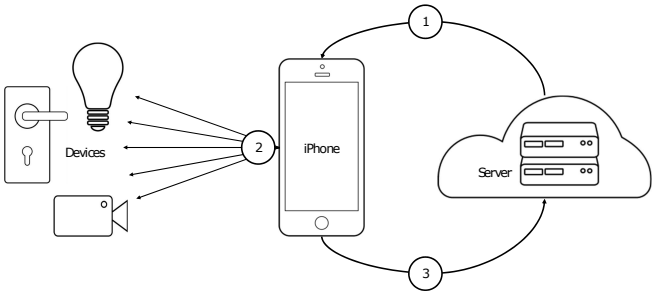


Figure 2: Malware scanning for IoT devices

3.2 Scouting for IoT Devices

Fig. 2 shows the steps involved in scouting for IoT devices in the home. In step ① the malware is remotely triggered. In our case, the trigger happens when the server sends to the mobile app a survey containing the keyword "IoT Devices", as illustrated in Fig. 1.

Once triggered, in step ② our malware scans for IoT devices in the home. There is no "one" protocol for discovering IoT devices. A multitude of standards exist, including UPnP, Alljoyn, Bonjour, and IoTivity amongst others. For illustration, in this paper we have chosen to focus our efforts on UPnP, since it is also the most widely implemented. UPnP is a package of network protocols that allow devices to quickly and automatically establish their presence in the network. Devices which implement UPnP are designed to be able to interface with other networked devices straight out of the box with minimal configuration.

Simple Service Discovery Protocol (SSDP) is the protocol adopted by UPnP that facilitates the automatic discovery and identification of devices connected to the local network. Its widespread use by devices, including those which do not implement any other UPnP technologies, makes it an ideal choice for our malware to search for devices on the local network. We note that SSDP discovery on the home network can only be performed from within the home network; hence the app on the user's smart-phone can do so, but not any external entity on the Internet. SSDP searches are initiated by sending a multicast search packet (MSEARCH) over UDP to the multicast address `239.255.255.250` with default port 1900 as assigned by the Internet Assigned Numbers Authority (IANA), and as shown in Fig. 3. Devices reply to MSEARCH packets by sending a packet with basic

information. A URL to a device description file is included in the packet under the location tag.

```
M-SEARCH * HTTP/1.1
HOST:239.255.255.250:1900
MAN:"ssdp:discover"
ST:ssdp:all
MX:3
```

Figure 3: An example of the body of an MSEARCH packet; to discover all devices we set the search target ST to ssdp:all

Our malware scouts for devices over both WiFi and bluetooth low energy (BLE). BLE devices send an advertisement packet periodically. Depending on the device the period between advertisements can be as short as 20ms or as long as 10.24 seconds. Other devices scanning for BLE devices can respond to these advertisement packets in order to connect and learn more information. As advertisements occur with a max period of 10.24 seconds, an 11-second interval sufficiently captures all BLE devices in the local environment. Since this is also a reasonable amount of time for the WiFi scouting to complete, we utilize the 11-second mark as a convenient time to halt the malware's discovery process.

In step ③, our malware packages and uploads the responses to the external server. We handle each response packet on a rolling basis. For each packet we locate the associated device description xml file and parse it to build a dictionary of device metadata and services offered by that device. Once parsing of the xml is finished we upload the dictionary as a JSON string to our server via an HTTP POST request. Examples of devices our malware discovers include light-bulbs, webcams, power-switches, and fitbits; specific devices of interest will be demonstrated in §4.

3.3 Attacking IoT Devices

Once our malware discovers the IoT devices in the home and reports them to the external server, an attack on any chosen IoT device can be initiated, following the steps shown in Fig. 4. In step ④, the server instructs the malware, using an HTTP GET request, on the parameters of the desired port-mapping so it can directly access the victim IoT device across the home router. The port-mapping mechanism configures the home router to map an incoming packet from the Internet, addressed to a specific transport-layer port at the home's public IP address, to a specific private IP address and port within the home; in other words, it allows specific unsolicited Internet traffic to enter the home.

Equipped with the parameters of the port-mapping, in step ⑤ our malware issues a UPnP command to set the desired port-mapping on the home router. Most off-the-shelf routers by default support automatic port-mapping via UPnP, in order to allow services such as peer-to-peer file sharing, user-hosted game servers, and video calling to function automatically without requiring manual configuration. Unfortunately, it is also what makes our attack vector a serious security threat for IoT devices. The UPnP protocol stack has no in-built security mechanism, and allows any host on the local network to issue commands without any authentication [13]; though there have been efforts to secure UPnP [20], such extensions are not implemented on common residential gateways (we have tried models from TP-LINK, Linksys and Netgear). Our app is therefore able

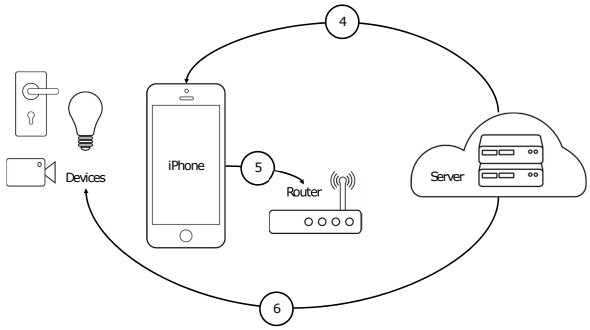


Figure 4: Malware setting up IoT attack

to create arbitrary port-mappings on the residential gateway. Our malware identifies the local IP address and port number of the home router from its earlier scan by looking for the `WANIPConnection` service that identifies a router. It therefore directs its port-mapping UPnP command to it, using a SOAP-based command sent over HTTP, as shown in Fig. 5. Here `remoteHost`, `externalPort`, `internalPort` and `internalClient` are the port-mapping parameters the malware obtains from our attack server.

```
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding"\>
<s:Body>
<u:AddPortMapping xmlns:u="urn:schemas-upnp-org:
service:WANIPConnection:1"\>
<NewRemoteHost>{remoteHost}</NewRemoteHost>
<NewExternalPort>{externalPort}</NewExternalPort>
<NewProtocol>{protocol}</NewProtocol>
<NewInternalPort>{internalPort}</NewInternalPort>
<NewInternalClient>{internalClient}</NewInternalClient>
<NewEnabled>{enabled}</NewEnabled>
<NewPortMappingDescription>{mappingDescription}
</NewPortMappingDescription>
<NewLeaseDuration>{leaseDuration}</NewLeaseDuration>
</u:AddPortMapping>
</s:Body>
</s:Envelope>
```

Figure 5: The structure of an UPnP WANIPConnection AddPortMapping command.

Once the port-mappings have taken effect, the external server has direct access to the IoT device in the home. Depicted as step ⑥, it can now attack the device to exploit known vulnerabilities, such as poor authentication credentials. As we will demonstrate in the next section, these vulnerabilities which were earlier limited to the home network, have now been exposed to the Internet, allowing an outside entity to take control of the smart-home in spite of the perimeter security provided by the home router.

4. EXPERIMENTAL EVALUATION

4.1 Setup and Default Behavior

Our experimental setup is shown in Fig. 6. We emulate a home environment in our lab, comprising two IoT devices: a D-Link DCS5300G camera and a Belkin WeMo switch. Both have known vulnerabilities, in that they lack authentication

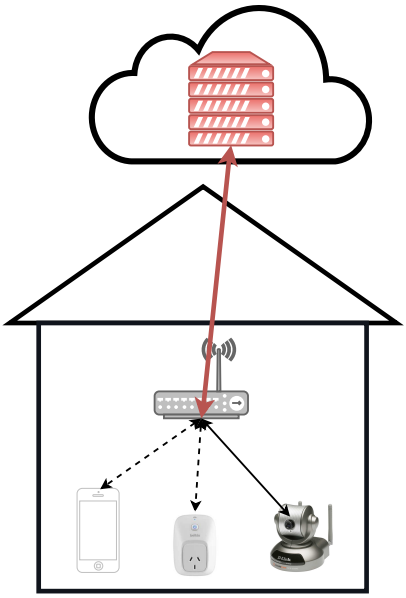


Figure 6: Experimental setup showing the cloud-based attack server, a Netgear R7000 wireless router, an iPhone, a Belkin WeMo switch and a D-Link camera.

credentials. The D-Link Internet camera comes out of the box with no credentials required to access the device, and similarly the WeMo switch will accept a `SetBinaryState` on/off command from any entity over its command port [15].

The IoT devices connect to the Internet via an off-the-shelf home router, the Netgear Nighthawk R7000 Wireless router in our case. The router is operated with default configurations, and assigns private IP addresses to the IoT devices. We operate an attack server in the cloud, that has scripts for attacking both IoT devices. We verified that attacks from the server do not reach the IoT devices (even if the server has all required information about the devices), since the home router by default drops all unsolicited incoming packets.

4.2 Enabling the Malware

We downloaded the survey app "Loki", containing the malware, from the Apple AppStore and installed it on an iPhone. We emphasize that no subjects other than the researchers in this project downloaded the app, and it was rapidly withdrawn from the AppStore once we had verified its functionality. We then created a survey questionnaire on the app server that contained our trigger phrase, as shown in Fig. 1. We then took the trigger survey on the iPhone, which caused our malware to begin scouting for WiFi and BLE devices in the home. We could see the HTTP POST messages coming to the server. Our server takes the body of these requests with the information of the discovered devices and stores it in a database. This is reflected by the server's front-end webpage, as shown in Fig. 7 depicting the uuid of the devices seen in the scans. Fig. 7(a) shows the D-Link camera details, including the manufacturer, model number and description, public IP address of the house, and the URL of the device including its private IP address. Fig. 7(b) shows similar details of the Belkin WeMo switch. We tested the malware in multiple homes, and found the reconnaissance by the malware to detect many kinds of devices, including the home router itself, which has the `WANIPConnection` device-type. This ability to discover household devices would have been nearly impossible without the infiltration of the home by the malware.

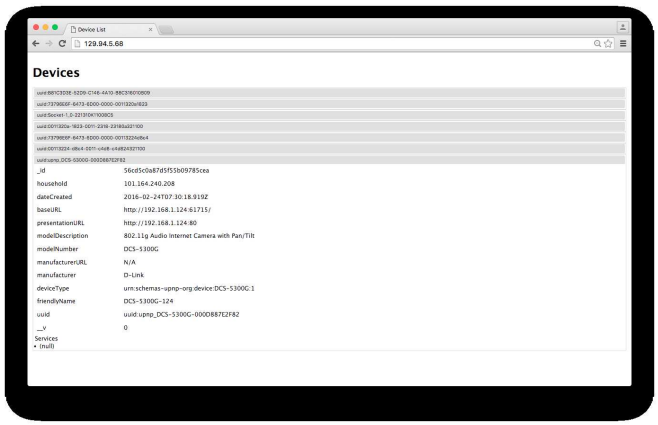
4.3 Attacking the IoT Devices

Once the malware thread in the app has completed the discovery process, it begins sending GET requests to the server to fetch port-mapping parameters in order to punch holes through NAT and expose the selected local devices. In our experiment the server, having detected the WeMo switch amongst the detected devices, instructs the malware to map traffic to port 49154 (the port on which the WeMo is listening to commands, as shown in Fig. 7(b)) to Internal IP address 192.168.1.128 (the private address of the WeMo switch). The malware sends a UPnP message to the home router to do so, and this succeeds. Thereafter, the attack server is able to send appropriately formatted commands to the home's public IP address on port 49154, which get forwarded by the router to the WeMo switch. Since the WeMo switch does not implement any authentication, the attack server is able to control it remotely, turning it off and on at will. Similarly, we were able to instruct the malware to configure port-mapping on the home router to redirect traffic from our attack server addressed to port 80 to the D-Link DCS5300G Internet Security Camera. With this, the camera's web interface was available via the Internet, and the attack server was able to exercise full control over the camera.

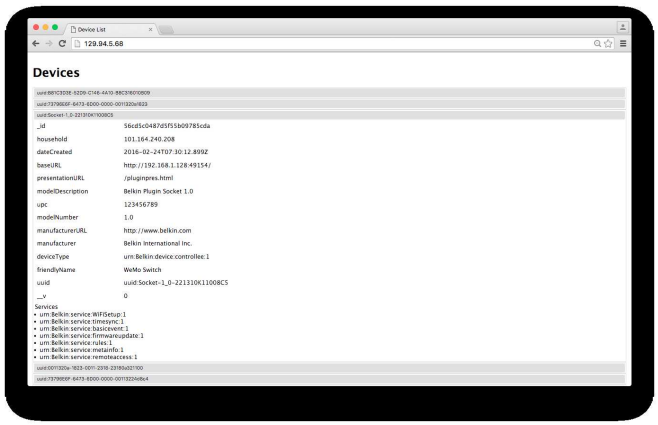
5. DISCUSSION AND CONCLUSIONS

We have demonstrated in this paper that it is possible to release malware-laden smart-phone apps that can circumvent the firewall protection offered by home routers. Specifically, the malware can scout the home network for IoT devices, and expose them at will to external attack. This has far-reaching implications. An attacker can use such malware to build a database of household IoT devices, while also creating port-mappings on the home routers in readiness for a future attack. An attacker can thus launch a large-scale attack against these households at a time of their choosing, or worse yet, offer this as a service to other malicious entities. In some ways this parallels the large-scale DDoS attacks prevalent today (such as the DD4BC extortion scheme [4]) that abuse the SSDP, DNS, and NTP protocols to amplify attacks on victims, with significant economic costs.

Fixing the security problems demonstrated in this paper is not easy. Security extensions to the UPnP protocol, though available [7], are unlikely to be implemented by home router manufacturers, since their incentive is limited to making it simple for non-technical users to run peer-to-peer applications and game-servers that need to discover network presence and establish network services. Screening mobile Apps to identify malware is also non-trivial, since Apps may legitimately access UPnP services, or change behavior via a trigger (like ours) once they have passed through the screening process. Ideally, IoT device manufacturers should be embedding better security in their devices and reducing reliance on perimeter security; however, this may take a long time to eventuate. In the intermin, it might be worthwhile investing in security solutions that analyze network traffic to deduce illegitimate access, along the lines of the proposals in [21, 22].



(a) D-Link webcam



(b) WeMo switch

Figure 7: Detecting the (a) D-Link webcam and (b) WeMo switch inside the home

6. REFERENCES

[1] . IEEE P2413 Standard for an Architectural Framework for IoT. http://grouper.ieee.org/groups/2413/Intro-to-IEEE-P2413.pdf.

[2] . M2I Security Framework. http://www.m2isf.com/.

[3] . Online Trust Alliance. https://otalliance.org/.

[4] . DD4BC Group Targets Companies with Ransom-Driven DDoS Attacks. http://www.tripwire.com/state-of-security/security-data-protection/cyber-security/dd4bc-group-targets-companies-with-ransom-driven-ddos-attacks/, Jun 2015.

[5] . Google's first Brillo and Weave partners introduced at CES. http://www.digitaltrends.com/home/google-iot-brillo-weave-partners/, Jan 2016.

[6] Arxan Technologies. State of Application Security Report. https://www.arxan.com/wp-content/uploads/2015/06/State-of-Application-Security-Report-Vol-4-2015.pdf, Jun 2015.

[7] C. Ellison. UPnP Device Security: Service Template. http://upnp.org/specs/sec/UPnP-sec-DeviceSecurity-v1-Service.pdf, Nov 2003.

[8] Claud Xiao. More Details on the XcodeGhost Malware and Affected iOS Apps. http://researchcenter.paloaltonetworks.com/2015/09/more-details-on-the-xcodeghost-malware-and-affected-ios-apps/, Sep 2015.

[9] ExtremeTech. Philips Hue LED smart lights hacked, home blacked out by security researcher. http://www.extremetech.com/electronics/163972-philips-hue-led-smart-lights-hacked-whole-homes-blacked-out-by-security-researcher, 2013.

[10] Forbes. Baby Monitor Hacker Still Terrorizing Babies And Their Parents. http://www.forbes.com/sites/kashmirhill/2014/04/29/baby-monitor-hacker-still-terrorizing-babies-and-their-parents/#7784ae4817e2, 2014.

[11] Gartner. Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015. http://www.gartner.com/newsroom/id/2905717, Nov 2014.

[12] S. Grover and N. Feamster. The Internet of Unpatched Things. In *Proc. FTC PrivacyCon*, Jan 2016.

[13] A. A. M. M. Haque. UPnP Networking: Architecture and Security Issues. In *Proc. TKK Seminar on Network Security*, Nov 2007.

[14] iControl. State of the Smart Home. http://www.icontrol.com/docs/pdf/2014_State_of_the_Smart_Home_-_Final.pdf, 2014.

[15] Isaac Kelly. Hacking the WeMo Switch. https://github.com/issackelly/wemo, 2012.

[16] NetworkWorld. 500,000 Belkin WeMo users could be hacked; CERT issues advisory. http://www.networkworld.com/article/2226371/microsoft-subnet/500-000-belkin-wemo-users-could-be-hacked--cert-issues-advisory.html, 2014.

[17] B. News. Fridge Sends Spam Emails as Attack Hits Smart Gadgets. http://www.bbc.com/news/technology-25780908, 2014.

[18] Nokia. Threat Intelligence Report. http://resources.alcatel-lucent.com/asset/193174, H2 2015.

[19] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli. An Experimental Study of Security and Privacy Risks with Emerging Household Appliances. In *Proc. International Workshop on Security and Privacy in Machine-to-Machine Communications (M2MSec)*, Oct 2014.

[20] T. Sales, L. Sales, H. Almeida, and A. Perkusich. A UPnP extension for enabling user authentication and authorization in pervasive systems. *Journal of the Brazilian Computer Society*, 16(4):261–277, Nov 2010.

[21] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani. Network-Level Security and Privacy Control for Smart-Home IoT Devices. In *Proc. IEEE WiMoB Workshop on Internet of Things Communications and Technologies (IoT-CT)*, Oct 2015.

[22] T. Yu, V. Sekar, S. Sheshan, Y. Agarwal, and C. Xu. Handling a Trillion (Unfiable) Flaws on a Billion Devices: Rethinking Network Security for the Internet-of-Things. In *Proc. ACM HotNets*, Nov 2015.