Cornelia Wolf Wilson
COMP 116: Introduction to Computer Security
Final Project, Fall, 2016

Pressing the Amazon Dash Button: A Case Study in Using the OWASP IoT Top Ten List to Analyze IoT Device Security

Abstract

In 2015, Amazon introduced the Dash button, a small, cheap wifi and bluetooth connected button that allows the Amazon customer to quickly reorder commonly used products. But, is this convenient shopping device also a convenient portal into the user's private data? What are the security implications of this inexpensive IoT device? In this paper, I will use OWASP's IoT Top Ten list and security testing guidelines to evaluate this product.

Introduction

The Internet of Things (IoT) is a rapidly growing industry that promises to make our lives better and easier by making the objects we interact with smarter. Broadly speaking, IoT refers to the addition of internet connectivity, sensors, and processing and networking capabilities to traditionally "dumb" tools. This increased connectivity and data flow can have impacts that range from the significant — think self-driven cars and better medical devices — to the mundane — e.g., a coffee machine you can turn on from your bed.

While "smart" devices can make our lives better, safer, and more convenient, they also open users up to a whole new world of security vulnerabilities. Because IoT is relatively new and often inexpensive, the security on many of these devices seems to have come as an afterthought, if at all. Because it is already relatively pervasive, however, these vulnerabilities can significantly impact its often unwitting users.

To The Community

The topic of IoT security has thus become almost as big a topic as IoT itself. A Google Scholar search for "internet of things security" returns more than 1.2 million results. Even the federal government is involved; among other actions being taken, the FBI published a PSA in September, 2015 with the warning "internet of things poses opportunities for cyber crime."[1] Not only can weak IoT security expose sensitive data and open users up to other types of hacking, but these devices can also be exploited, without the user's knowledge, to perpetuate larger scale attacks. In October, 2016, for instance, the major distributed denial of service (DDoS) attack on the domain name system company Dyn that downed major sites in the US and Europe was carried out using thousands of insecure IoT devices as proxies.[2] OWASP, the Open Web Application Security Project, has a project dedicated to IoT security, headed by Daniel Messier and Craig Smith. According to its mission statement this project is "designed to help manufacturers, developers, and consumers better understand the security issues associated with the Internet of Things, and to enable users in any context to make better security decisions when building, deploying, or assessing IoT technologies."[3] In the vein of their popular top ten list of web application security risks, OWASP has released an IoT top ten list as well as a testing guide for IoT technology.

On the spectrum of severity of attacks against users, connected medical devices are clearly at one extreme. Weak security on these devices could be exploited to access medical records or even compromise treatment. Discussing a vulnerability found in a Hospira Infusion Pump, the FDA statement recognized that "these vulnerabilities could allow unauthorized users

---

[1] https://www.ic3.gov/media/2015/150910.aspx

[2] https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[3] https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main

to control the infusion pump and modify the dosage it delivers, potentially leading to over- or under-infusion of critical patient therapies."[4] Vulnerabilities in connected motor vehicles are also extremely dangerous. The FBI, together with the Department of Transportation and the National Highway Traffic Safety Administration, issued a warning in March of 2016 that "motor vehicles [are] increasingly vulnerable to remote exploits," which could allow an attacker to gain control of vehicle functions, including brakes and steering.[5]

On the other end of this spectrum might lie Amazon Dash buttons. These are small, cheap Wi-Fi and bluetooth enabled buttons that allow Amazon customers to quickly reorder commonly used products. In one of the advertised use cases, for instance, the customer sticks a Tide branded Dash button to a washing machine and can reorder their favorite laundry detergent with one click.

Generally, these buttons make a simple task slightly easier, and do little else. As much as IoT promises to make hospitals smarter and roads safer, it also encompasses many little tools like this one that make ordinary actions more convenient and are desirable in part for their novelty factor. Still, these five dollar buttons connect to the internet via your wireless router and are linked, of

---

[4] http://www.fda.gov/MedicalDevices/DigitalHealth/ucm373213.htm

[5] https://www.ic3.gov/media/2016/160317.aspx

course, to your Amazon account, so they could potentially expose the user's data or be open to exploits like those used in the DDoS attack mentioned above.

Applications/Action Items

Ultimately, and unfortunately for this project, I think the worst problem with the Dash button is that you'd have to see another Tide advertisement every time you do the laundry, and the biggest danger is that you'll accidentally order 500 rolls of toilet paper (although Amazon makes this difficult to do). Nevertheless, I took it as an interesting test case and illustration of the OWASP IoT testing guidelines. These guidelines are a first step in addressing the growing problem of IoT security. They can be used both by developers to secure the products they build and by customers to understand the security implications of the devices they use. As IoT grows, these recommendation should be refined and expanded. Moreover, security should be incorporated into the design of products and not included as an afterthought. Consumers might be able to shape this trend by educating themselves with resources like this and purchasing devices that fulfill basic security screening.

In introducing their top ten list, OWASP emphasizes that a holistic approach must be taken when analyzing an IoT device.[6] In focusing on the device itself, you may miss other surface areas that should be evaluated. Keeping this in mind, the 2014 OWASP Internet of Things Top Ten is as follows:

- I1 – Insecure Web Interface
- I2 – Insufficient Authentication/Authorization
- I3 – Insecure Network Services

[6] https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf

- I4 – Lack of Transport Encryption
- I5 – Privacy Concerns
- I6 – Insecure Cloud Interface
- I7 – Insecure Mobile Interface
- I8 – Insufficient Security Configurability
- I9 – Insecure Software/Firmware
- I10 – Poor Physical Security

In the rest of this paper, I will discus each of these vulnerability areas in turn, illustrating the testing strategy on the Amazon Dash Button, where possible.

I1. Insecure Web Interface

This category deals with device administration interfaces. For many IoT devices, a web interface may be used in the initial setup or to control administrative settings. Such sites might be vulnerable to Cross-Site Scripting (XSS), Cross Site Request Forgery (CSRF), and SQL injection. They also might have poor credential management. This could include using default passwords initially, allowing weak passwords, not having an account lockout mechanism, or transmitting usernames and passwords insecurely.

In the case of the Amazon Dash button, the initial setup is performed via the Amazon shopping application for mobile devices, and can not be done over a web interface. The interaction with the mobile interface is discussed below.

I2. Insufficient Authentication/Authorization

This category has some overlap with the entries involving insecure interfaces — I1, I6, and I7 — but applies to all device services. The most significant problems that fall under this heading are weak password requirements, plaintext or poorly encoded transmission of credentials, insecure password recovery mechanisms, and misconfigured privileges. You can test

a devices for these weaknesses by attempting to set a weak password and examining packets sent over the network (using a tool like Wireshark) for passwords sent in the clear or encoded in base64.

Credentials management for the Dash button piggy-backs on the security that protects your Amazon account. Surprisingly, however, the only password requirement Amazon enforces when setting up an account is a 6-character minimum length. In fact, I was able to create an account using simply "password" as the password. Amazon does offer two-factor authentication that can be set up under Your Account > Change Account Settings > Advanced Security Settings.

As far as credential transmission is concerned, sniffing via Wireshark revealed that all the data transmitted to and received from the Amazon servers when the button is pressed is encrypted. I wasn't able to inspect the data sent over bluetooth between the button and whatever mobile device is used during the initial setup, but documentation on the Cypress BLE chip used states that it supports link-layer encryption.[7]

I3. Insecure Network Services

This entry mainly pertains to vulnerabilities stemming from open ports and the use of outdated, insecure protocols and services. If these are present, the IoT device may be opened up to buffer overflow exploits, fuzzing, and denial-of-service attacks.

An nmap scan of the IP address associated with the button (found via sniffing on the network when the button was pressed) didn't return any useful information; all ports were classified as closed. The button also doesn't respond to pings. Penetration testing tools could be used to further analyze the device for this type of vulnerability. Additionally, the button only

[7] http://www.cypress.com/file/139841/download

powers on when pressed, and only stays on long enough to send an ARP request and transmit a message to Amazon, which would make them unlikely candidates for use in DDoS attacks.

I4. Lack of Transport Encryption

This item has overlap with the entries regarding insecure interfaces (I1, I6, and I7) and with I2. As mentioned above, all data transmitted over the wireless network appears to be encrypted using the TLSv1 protocol, and the bluetooth chip uses link level encryption (although this has not been verified). According to Amazon documentation, all its IoT devices connect to AWS IoT using X.509 certificate-based authenticated connections, and end-to-end encryption is employed for all data transmission.

I5. Privacy Concerns

To assess how an IoT device addresses privacy concerns, we need to ask the following: Is unneeded personal data collected, and can the user control the types of collected data? Is personal data encrypted for storage and transmission? Is collected data anonymized? Who has access to collected data?

Setting aside the user-submitted and usage data collected via Amazon when setting up and using an account, the Dash button specifically may gather and transmit IP addresses, available Wi-Fi networks, and your network password. It also sends information about the device state to Amazon. The fact that you have to enter your Wi-Fi password into the Amazon mobile application is probably the most concerning part of using the dash button (save accidentally pressing it), but you can choose whether or not Amazon will save this password for future use.

I6. Insecure Cloud Interface

Similar to I1, this entry covers vulnerabilities in APIs and cloud-based web interfaces. OWASP recommends that all interfaces be reviewed for known vulnerabilities. These interfaces should also require strong passwords and possibly use two-factor authentication. This is not relevant to the Amazon Dash button.

I7. Insecure Mobile Interface

Mobile interfaces for IoT devices should follow all the same credential management precautions outlined under I1: requiring strong passwords, including a lockout mechanism, avoiding default usernames and passwords, and forcing password expiration after a specific period. Like the web-based interface, the mobile interface should also allow for two-factor authentication and use transport encryption when sending data.

Setting up the Amazon Dash button requires you to have a mobile device with the Amazon shopping app installed. The credentials management is the same as for the Amazon web interface (discussed above under I2), except that the mobile app can take advantage of other authentication methods native to the device (e.g. Apple's Touch ID). The user must be logged into the application with his or her Amazon username and password in order to initiate the setup, which is done via bluetooth (or ultrasound, if bluetooth is not available). During the setup process, the user has to select a Wi-Fi network for the button to use and enter the password into the app if necessary. The list of available Wi-Fi networks is transmitted to Amazon, and the password given may be stored, encrypted, on Amazon's servers if this option is selected.

I8. Insufficient Security Configurability

OWASP recommends that IoT devices include basic security configurability. With regards to credentials, the user should be able to opt into two-factor authentication and choose to use long passwords (more the 20 characters). The device should also have a granular permissions model, such that the administrative user can limit the capabilities of non-admin users. The user should also be able to choose to have their data encryption during storage and transmission, if this is not done automatically. Finally, the device's functionality should include security monitoring and logging, and should notify the user of any security events.

This category is especially important for devices with multiple users or multiple remote interfaces. Even if the device does not explicitly allow multiple accounts, permission escalation attacks might be possible on devices built on top of traditional operating systems like Windows or Linux. In the case of the Dash Button, the most relevant issue is security monitoring and logging, which does not seem to be available. The fact that the button can be used (i.e. pressed, resulting in an ARP request being sent to the router) without being properly initialized is something that might be disallowed with more robust security monitoring. This irregular usage is exploited in hacks for the button (discussed below).

I9. Insecure Software/Firmware

This topic deals mainly with the security of updates. Devices that automatically download updates that are not cryptographically signed or are transmitted over insecure channels (e.g. HTTP) are vulnerable to attacks via forged updates containing malicious code. Sensitive data may also be hard-coded into updates, so transmitting or storing them without encryption can also be dangerous. As far as I can tell, there's no mechanism in place for updating the software or

firmware of the buttons. The lifespan of the buttons are limited, however, by the life of the non-replaceable alkaline battery, which is said to be about 1,000 clicks. The buttons are also so inexpensive ($5, with a $5 credit towards the second purchase made with them) that simply replacing them would probably be more efficient than updating them.

I10. Poor Physical Security

The final entry in the OWASP list chiefly has to do with exposed physical ports, such as USB ports and headers. Devices with exposed ports should protect the OS by minimizing accessibility, disable unused ports, and limit administrative capabilities to a local interface. Regarding physical security, OWASP also warns that it should be difficult to disassemble the device and access or remove data storage mediums.

Other than the button itself and the small speaker opening (for using ultrasound during setup), there are no physical access-points on the device. Breaking open the plastic shell reveals the following: an alkaline battery, an Atmel ATSAMG55J19A-MU ARM micro controller, an Atmel ATWINC1500B wireless chip, a Cypress CYBL10563-68FNXI Bluetooth Low Energy chip, a Micron N25Q032 flash memory chip, and an LED light.[8]

Conclusion

In terms of security, this little device is quite innocuous. Both because it piggy-backs on the security used to protect Amazon accounts and transactions and also because its functionality is so limited, it poses little risk to the user. Aside from information already required to use Amazon, the most sensitive data the device requires is the name and password or your home Wi-

[8] https://mpetroff.net/2016/07/new-amazon-dash-button-teardown-jk29lp/

Fi network. In one sense, this is an example of a well designed and implemented IoT device: it requires only one interface (incorporated into the Amazon mobile app), performs a single function, collects only minimal necessary data, and only powers on briefly to perform the function. It also uses no physical ports, has a limited life-span (about 1,000 clicks, or the life of the battery), and is cheap and thus easily replaceable. This security is possible, however, because the device's capability is so limited. Despite fulfilling the OWASP's top criteria for a secure IoT device, this example is not adequate to fully understand OWASP's recommendations because the device itself doesn't reflect the possibilities and power inherent in IoT.

Addendum: Hacking the Amazon Dash Button

While my analysis of the Amazon Dash button failed to reveal any major security vulnerabilities, the device can still be "hacked" for unintended purposes. By leaving the setup step incomplete, you use the button to transmit a "message" to your computer or another device capable of sniffing on the network without purchasing anything. This "message" — which is really an ARP request from the button's known MAC address — can signal the receiving device to initiate another event, such as turning on a connected lightbulb or entering data into a spreadsheet.[9] In a YouTube video entitled "Amazon dash button automation silliness," for example, Michael Donnelly demonstrates how he programmed a dash button to turn down the AC in his Tesla.[10] Brian Jacobel has shared code on github that turns the button into a cyber-piggybank by transferring money from checking into a Simple savings account every time it's pushed.[11] This project is interesting because it uses a Raspberry Pi as the intermediary. Many other examples of hacks using the same methodology can be found online.

The core of the hack is the fact that the button powers on and transmits a sequence of messages every time it is pushed. First, an ARP request is sent to the router with the button's IP and MAC addresses. Soon after, the button begins exchanging data with the Amazon servers — in my case, this was at 54.239.25.216. A TCP Handshake is performed and a TLSv1 encrypted connection in initialized. The button first sends a "Client Hello" packet, which is followed by the

---

[9] The most cited (and perhaps first) discussion of the hack was published by Edward Benson on medium.com: https://medium.com/@edwardbenson/how-i-hacked-amazon-s-5-wifi-button-to-track-baby-data-794214b0bdd8#.blqyjynd5

[10] https://www.youtube.com/watch?v=e1hmjyNwrCQ&feature=youtu.be

[11] https://github.com/bjacobel/piggydash

key exchange. A single application data packet (of length 348) is then sent from the button, and three application data packets are sent from the Amazon IP in response (see figure 1). This process appears to be repeated, but the packet sent from the button is much longer (1260) and the Amazon server responds with four packets.

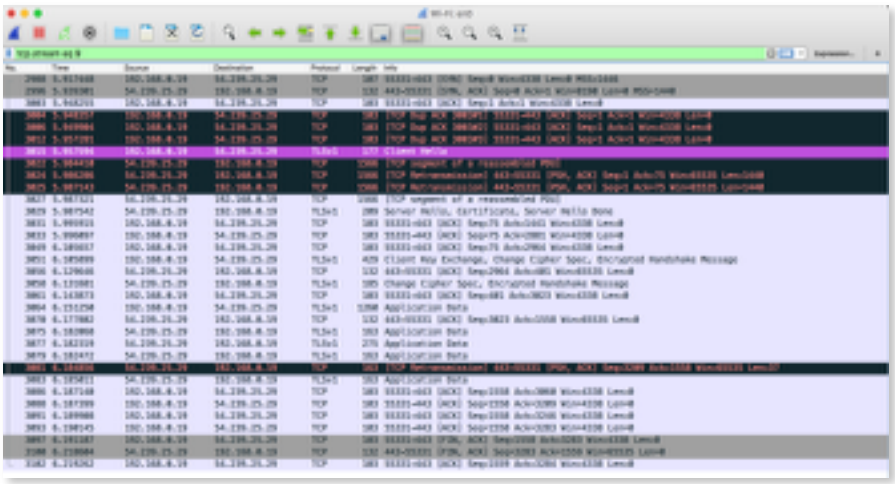

figure 1. Packets sent between the Amazon Dash button and the Amazon Servers in one session

Using a simple Python script and the Scapy libraries, you can filter for ARP messages sent from a specific button's MAC address and initiate an action once one is received. The MAC address can be found by using a tool like Wireshark to capture packets transmitted over the

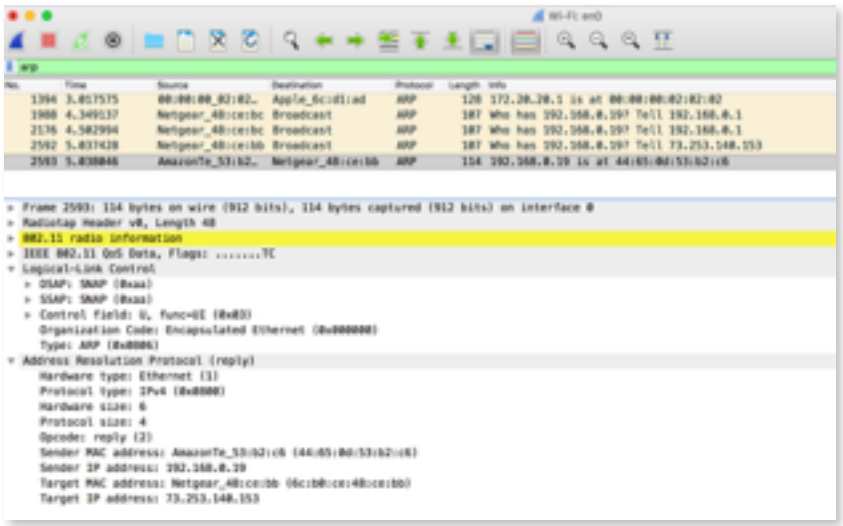

figure 2. analyzing ARP request to find button IP and MAC addresses

network and analyzing them. My Dash button, for instance, had MAC address 44:65:0d:53:b2:c6 (resolved to "AmazonTe_53:b2:c6") and was assigned IP address 192.168.0.19. Using Scapy's sniff command, your device will listening for ARP request packets and call the function "process_arp" with the packet as a parameter when one is received:

```
sniff(prn=process_arp, filter="arp", store=0)
```

The `process_arp(pkt)` function then checks the MAC address of the packet's sender against the known button address and calls another function, which can do anything you want.

Other Resources

Amazon.com. "Privacy Notice." Accessed December, 2016. https://www.amazon.com/gp/help/customer/display.html?nodeId=468496

Amazon.com. "Set Up Your Dash Button." Accessed November, 2016. https://www.amazon.com/gp/help/customer/display.html?tag=networkworld0a-20&nodeId=201746340

Amazon.com. "How the AWS IoT Platform Works." Accessed December, 2016. https://aws.amazon.com/iot/how-it-works/

OWASP. Internet of Things Project page. Last modified August 10, 2016. https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=Main

OWASP. "Internet of Things Top Ten." Accessed December, 2016. https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf

OWASP. "Tester IoT Security Guidance (Draft)." Last modified May 14, 2016. https://www.owasp.org/index.php/IoT_Testing_Guides

Jonathan Lampe. "How to Test the Security of IoT Devices." Published November 10, 2014. http://resources.infosecinstitute.com/test-security-iot-smart-devices/