An Analysis of Security Measures in IoT Devices

Jayanth Rajakumar
45243589
University of California, Irvine

Andrew Muzik
87404144
University of California, Irvine

Deyuan Wang
89048350
University of California, Irvine

Abstract

As web-connectivity becomes standard on everyday technologies, their security measures should not be overlooked. Encryption is not always employed, UDP/TCP ports can be left open for no good reason, and many times security measures are ignored altogether to lower manufacturing costs. This report will look into common IoT products and their susceptibility to exploitation by attackers, including a comparison of their security measures employed. It primarily focuses on two products, the Xiaomi Wifi Smart Plus and the KGuard QRT-502 camera. Various attacks such as phishing, DDOS and attacks on availability are discussed. Finally, various methods of mitigating the impacts of the vulnerabilities are considered.

1 Introduction

The field of IoT security is broad, taking principles from cryptography, networking, architecture, web development, and hardware. In each area, lies an attacker's malicious piece of code meant to exploit flaws in some specific IoT device. One of the most common devices in IoT, the webcam, possesses many vulnerabilities from the nature of the device - and a common target of attackers.

This paper will focus on the security of a KGuard QRT 502 webcam, readily available on the Amazon marketplace. This webcam is susceptible to a variety of attacks this paper will conduct to analyze their feasibility and mitigations. The KGuard family of webcams utilize a minified version of the Linux OS with multiple different methods to obtain the firmware of the QRT 502 device, further explained in the methodology.

A paper titled "SoK: Security Evaluation..." by Alrawi and Omar cover various attacks on different IoT devices, from bluetooth speakers to webcams, including leveraged trust from mobile applications, insecure cloud deployment, flaws in firmware code, and weak or even default authorizations [2]. Alrawi and Omar provide a case study of 45 devices and classified their security vulnerabilities.Some various security flaws exposed by Alrawi and Omar include reading an exposed pin on the device to gain root access, a Sonos device runs unauthenticated services on high ports and stealing WiFi credentials through firmware flaws [2].

Common mitigations include device updates for authentication and vulnerable service issues, while new frameworks are designed to mitigate inherent flaws in the platform. Several challenges were faced by the Alrawi team including encrypted iOS applications and automated device updates that may need to be taken into account [2].

2 Related Work

Many of the big IoT makers provide SDKs for their platforms to fast track product development by 3rd party vendors. Xiaomi takes a different approach, by using proprietary software and manufacturing their products from the ground up [5]. This closed-sourced platform may not prove a fruitful approach, as open-source software generally provides a higher level of scrutiny by 3rd party developers.

A variety of attacks exist on the Xiaomi platform of IoT devices, both theoretical and practical. One vulnerability Zhou et al. make note of is the variability of manufactures MAC addresses, as consecutive addresses are typically assigned to products of the same family, allowing for an easy brute force approach to the last 3 bytes [5].

Some various security flaws exposed by Alrawi and Omar include reading an exposed pin on the device to gain root access, a Sonos device runs unauthenticated services on high ports and stealing WiFi credentials through firmware flaws. Common mitigation techniques include device updates for authentication and vulnerable service issues while new frameworks are designed to mitigate inherent flaws in the platform . Several challenges were faced by the Alrawi team including encrypted iOS applications and automated device updates that may need to be taken into account. Similar authentication vulnerabilities and challenges to Alrawi's research are seen in this paper. [2]

To connect the KGuard webcam, a client connects to the device's network and configures the device to the client's network. Xiaomi's family of IoT devices use encryption to configure the device upon booting up to bind the cloud-device connection. Xiaomi's use of encryption poses a challenger to attackers, however security breaches tend to be well documented for additional attackers to exploit [5]. KGuard appears to also use network level encryption, leading to the challenge of reverse engineering their binding process without the ability to sniff packets in the local network.

3 Methodology

In this section, we consider two case studies of commonly available IoT products and describe their security features and vulnerabilities.

3.1 Xiaomi WiFi Smart Plug

This product is a wifi enabled pass-through plug that can turn on or off any electrical device powered from it through a smartphone app. It is part of the Xiaomi ecosystem of smart home products which consists of over 800 different models. [1]

The plug relies on the Xiaomi Cloud server to act as an intermediary between the itself and the user/app. The app uses OAUTH to authenticate users using a Xiaomi account.

When a user issues an on or off command from the app, a HTTPS message layered with AES is sent to the global Xiaomi cloud server. The server then responds directly to the plug with an AES-encrypted UDP packet in order to control it. The usage of HTTPS and AES increases the security of the system, and offers protection against unauthorized connections which may be disastrous if malicious attackers target sensitive devices.

The device is also secure from a networking perspective as there are no open TCP ports which could serve as targets for attack. However, the protocol used by the Xiaomi server to communicate with any Xiaomi device has been reverse engineered and custom Python scripts have been developed to send UDP packets directly to the device to control it [4]. The only requirement is the AES key which is based on a dynamic token ID. This token can only be obtained if an attacker has physical access to a smartphone that is already paired with the plug. As such, this is vulnerability is not very threatening and is unlikely to lead to any attacks.

3.2 QRT-502 Webcam

The QRT-502 is a network camera manufactured by KGuard Security. This section explores the security measures and vulnerabilities of the camera.

3.2.1 Overview

The camera is powered by an ARM-based processor and runs a minified version of Linux. The setup process involves switching the camera to Access Point mode, and connecting from a smartphone app to perform the initial configuration. From there it can be switched to connect to any other wifi network. The camera's live feed can be viewed from the app as well as a web portal running on port 80 at the local address of the camera. The portal and website also allow configuration changes such as wifi credentials, recording settings and image quality.

3.2.2 Reverse Engineering

This section describes our effort to reverse engineer the working of various systems of the camera in order to search for vulnerabilities.

First we performed a scan of all TCP ports on the camera using the NMAP utility. Three ports were found to be open and accepting connections:

1. Port 23: This port is used to provide a telnet backdoor for debugging purposes. It is secured by a hardcoded password.

2. Port 80: It is used to host the camera's web server to display the live feed as well as for configuration

3. Port 554: It serves a live RTSP (Real Time Streaming Protocol) stream from the camera. This feed is not secured and does not require any authentication.

The telnet backdoor presents an opportunity to gain root access to the system. In order to obtain the login credentials, we examined the firmware update binary of the camera that we obtained from the manufacturer's website. On scanning with the binwalk utility, we found a JFFS2 filesystem that we mounted on a Linux system.

This gave us access to the entire root directory of the camera, and we extracted the /etc/passwd file which contains the root username and hash of the password. We performed a dictionary attack on the hash using the John the Ripper utility and obtained the password in under 30 minutes. In this way, an attacker can obtain root access to the camera and perform a number attacks targeting the confidentiality, integrity and availability of the camera and users' data.

3.2.3 Vulnerabilities and Exploits

Next we explore some of the vulnerabilities of the camera, and discuss the attacks that take advantage of these vulnerabilities.

Weak default password The default login and password for the administrative interface are admin and admin respectively. This means that anyone connected to the same local

network as the device can gain access by simply scanning the network to find the camera's IP address. This will reveal details such as the wifi credentials and also allow them to change the admin password, wifi network, recording settings, etc.

No HTTPS The administration web page is not secure and uses plain HTTP. This means that any traffic related to this can be intercepted and potentially modified by anyone on the network. Sensitive content that may be leaked includes: Dynamic DNS (DDNS) credentials, wifi credentials and email credentials.

Telnet Backdoor There is a backdoor that allows root access to the camera's operating system. As described in the previous section, port 23 is open and accepts telnet connections. This gives an attacker root access to the camera and allows them to run practically any Linux commands to perform further attacks.

Phishing Attack Phishing attacks are social engineering attacks where a user is misled into entering their credentials into a page/window created by the attacker to look like a legitimate website. Attackers typically target users who are inexperienced with technology such as the elderly. We created a proof of concept of a phishing attack on the camera's web portal login page. image. When the attack is deployed, after logging in to the standard camera web page, the user is redirected to a fake Google login page hosted on the camera (Figure 1). If they believe that it is a legitimate page and enter their Google credentials, the attacker remotely captures them.

The overall picture of the attack is shown in Figure 2. First an attacker obtains root access via a telnet session. Then they download the files required for the attack from their own server on to the camera using the wget command. These files included modified camera login page and javascript file, as well as the fake Google login page. Now the attack is ready, and if a user becomes a victim of the attack, their credentials are transmitted to the attacker's remote server using a HTTP request.

SYN Flooding The camera has limited system memory resulting in susceptability to denial of service attacks. When a SYN flooding attack is performed using netwox 76, the camera stalls and is no longer accessible via the website. The camera then resets itself. This is in spite of the fact that the system has SYN cookies enabled. The netwox command to carry out this attack is shown in Figure 3. An attacker on the local network can simply send a SYN flood to disable the camera and then carry out malicious activities on the premises. As long as the SYN flooding is carried out, the camera will continue resetting with no audio or visual indication of malicious activity.

```
[12/14/19]seed@VM:~$ sudo netwox 76 --dst-ip 192.168.0.212 --dst-port 80
^C
[12/14/19]seed@VM:~$ arp -a
? (192.168.0.1) at 80:d0:4a:f9:b6:32 [ether] on enp0s3
? (169.254.215.225) at <incomplete> on enp0s3
? (192.168.0.212) at e0:b9:4d:aa:2f:e8 [ether] on enp0s3
? (192.168.0.214) at b4:ae:2b:d7:02:84 [ether] on enp0s3
[12/14/19]seed@VM:~$ sudo netwox 76 --dst-ip 192.168.0.212 --dst-port 80
```

Figure 3: Syn Flood Attack with Netwox 76 Tool and ARP for IP.

Permanent Inoperability Another attack on the availability of the camera is to write a script that is triggered on every startup. This can be done in the /etc/init.d directory. The rcS script is observed to execute on every start. An attacker can add his own shell commands here. For example, adding a command to reboot the camera on every startup makes it permanently inoperable.

Botnet If the camera is left exposed to the internet instead of being protected by a NAT, then they may be used to create a botnet. An attacker can scan the IP space to find publicly accessible instances of these cameras and inject a script which targets a third party server. For example, similar to the Mirai botnet, an attacker can compromise thousands of cameras and use them to ping flood a DNS provider. If the attack succeeds, large parts of the internet could be affected.

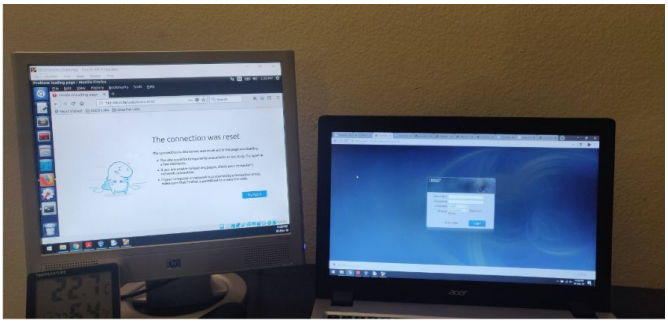

Figure 4: Camera is protected by TCP RST. The system on the list is not authorized to connect to the camera, so it is targeted by TCP RST attack and fails to connect to the camera's web login page.

3.2.4 Mitigation

The ideal solution is for the manufacturer to patch the exploits. Adding HTTPS and removing the telnet backdoor would address most of the security issues. However manufacturers usually do not provide timely software updates over the lifetime of a device.

Another solution could be for the local network to only allow specific known devices to connect to the camera. This is similar to VLANs seen in enterprise networks, however this feature is unlikely to be present in commercial grade routers.

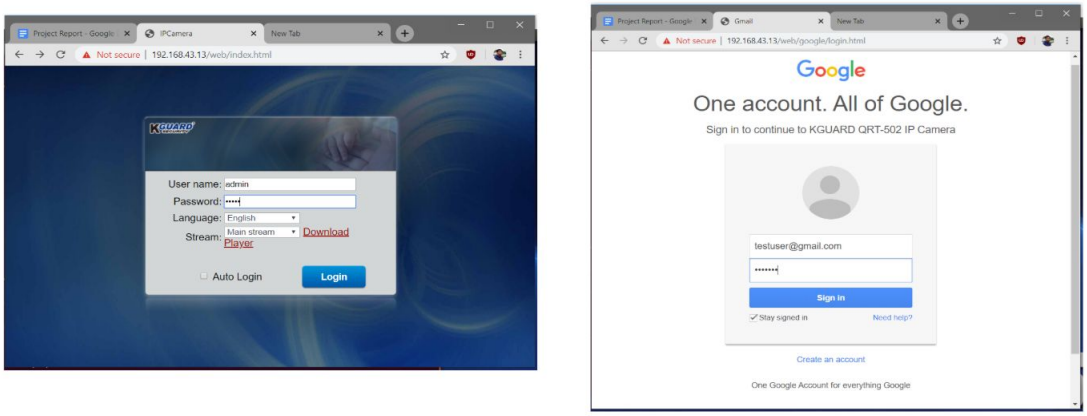

Figure 1: Phishing attack on the camera's login page. After performing the regular login, the user gets redirected to a fake Google login page.

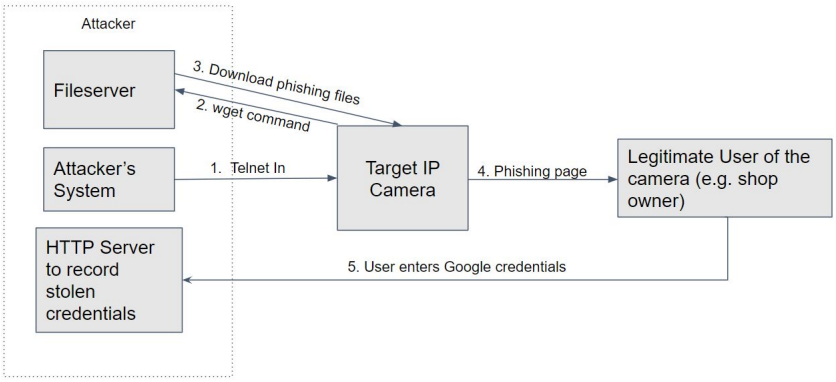

Figure 2: Overall picture of the phishing attack

An alternate solution is to use the TCP RST can be used as an attack to prevent unauthorised connections.This would use a system that is always running to sniff the traffic present on the local network, and send a TCP RST packet to any unauthorized device that tries to connect to the camera.

To implement this, we developed a script using pyshark, a Python implementation of wireshark. The script continuously sniffs packets on the local interface and checks the source and destination IP addresses. A specific list of whitelisted devices is specified. If a device not in the list tries to connect with the camera, then it issues a TCP RST attack on that device using the netwox utility. Figure 4 shows the result of using this type of mitigation.

4 Evaluation

Analysis of the implementations show successes but they aren't without drawbacks. Phishing attacks can be easily avoided through analyzing the insecure url. Syn flooding attacks can be remedied through enabling syncookies, but given the limited resources on many IoT devices, even if they're enabled they have a limited range of effectiveness in attack mitigation.

The phishing attack presented here is very dependent on naive user operation, but given the success of basic phishing attempts through email it was worth documenting here. Availability attacks are harder to prevent with the device left open to the attacker. Although syn flooding is an easy remedy through enabling and compiling with syncookies enabled, they are only effective on the most basic TCP handshake options and not designed to handle TCP options such as timestamps to handle video feed.

One possible 3rd party patch was implemented in this paper that used an access control methodology to block any unlisted IP from accessing a telnet communication. However, if an attacker were to spoof a victim IP address they would be able to break the mitigation. This mitigation technique remains theoretical and further research must be conducted.

5 Future Work

Additional research is needed in reverse engineering the binding process between client and device. Buffer overflows can be searched for using tools like Valgrind to be exploited. Additionally, the KGuard firmware libraries are not reinstalled between each initialization, leaving the device compromised to the attacker until a restart is performed on the device. Further attacks on availability to be investigated include the ping of death and smurf attacks.

The exploits discussed assume that the attacker is connected to the same local network as the camera. In real-life situations, devices are protected by a NAT, and an attacker cannot directly initiate a connection with the camera. Further research is required to understand the extent to which the vulnerabilities discussed pose a threat from an external attacker.

Research from [5] has shown a growing trend from major players in the industry such as Amazon and Samsung move away from proprietary IoT development toward open-source development frameworks. Just recently, Xiaomi has announced they too will be joining in arms for open-source development in the IoT space [3].

6 Code Implementation

The scripts and files used for the implementation of the phishing attack and TCP RST mitigation measure can be found at https://github.com/jayanth-rajakumar/EECS221Project.

References

[1] D. Giese. Having fun with IoT: Reverse engineering and hacking of Xiaomi IoT devices. In *DEFCON 26*, 2018. https://dgiese.scripts.mit.edu/talks/DEFCON26/DEFCON26-Having_fun_with_IoT-Xiaomi.pdf.

[2] M. Antonakakis O. Alrawi, C. Lever and F. Monrose. *SoK: Security Evaluation of Home-Based IoT Deployments*. IEEE Symposium on Security and Privacy, 1.00 edition, 2019. https://alrawi.github.io/static/papers/alrawi_sok_sp19.pdf.

[3] Xiaomi Team. Xiaomi amplifies commitment to open-sourced software. In *Xiaomi Corporation*, 2019. http://blog.mi.com/en/2019/07/21/xiaomi-amplifies-commitment-to-open-source-software/.

[4] R. Teemu. Python library and console tool for controlling xiaomi smart appliances. https://github.com/rytilahti/python-miio.

[5] Y. Tao W. Zhou, Y. Jia and Y. Mao P. Liu Y. Zhang L. Zhum, L. Guan. *Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms*. USENIX Security Symposium, 1.00 edition, 2019. https://alrawi.github.io/static/papers/alrawi_sok_sp19.pdf.