TrendLabs

Device Vulnerabilities in the Connected Home:

Uncovering Remote Code Execution and More

Technical Brief

TrendLabs Security Intelligence Blog
Dove Chiu, Kenney Lu, and Tim Yeh
Threats Analysts

April 2018

TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.

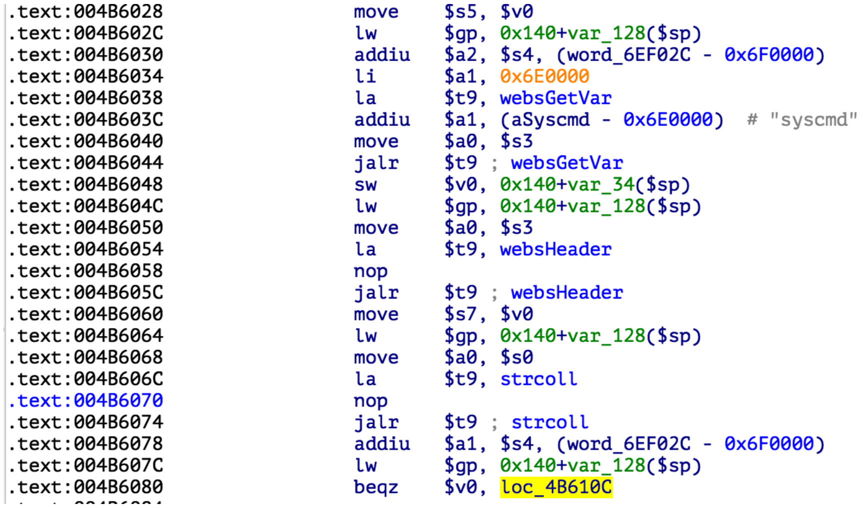
No	Manufacturer	Model	Vulnerability	ZDI/CVE/IPA
1	Belkin	Netcam HD+ WiFi Camera	Remote code execution	ZDI-18-132, ZDI-CAN-4970
2	Belkin	WeMo® LED Lighting Starter Set	Unauthorized SYSEVENT service	ZDI-18-133, ZDI-CAN-5095
3	Belkin	WeMo®	Denial of service	ZDI-18-134, ZDI-CAN-5206
4	Buffalo	WSR-300HP	Command injection	IPA#74871939
5	D-Link	DCS825L EyeOn Baby Monitor	Command injection	CVE-2017-11563
6	D-Link	DCS825L EyeOn Baby Monitor	Stack overflow	CVE-2017-11564
7	Dahua	IP Camera and PTZ Camera	Predictable recovery password	ZDI-18-130, ZDI-CAN-4956, CVE-2017-9315

Table 1. IoT devices found with vulnerabilities

1. Belkin Netcam HD+ WiFi Camera – Remote Code Execution (SSRF + LCE)

Belkin Netcam has a local code execution (LCE) vulnerability, which can execute arbitrary commands from localhost (on the device) via internal HTTP API. However, its access is limited, so an attacker cannot use it remotely.

Non-Public API (/bin/goahead)

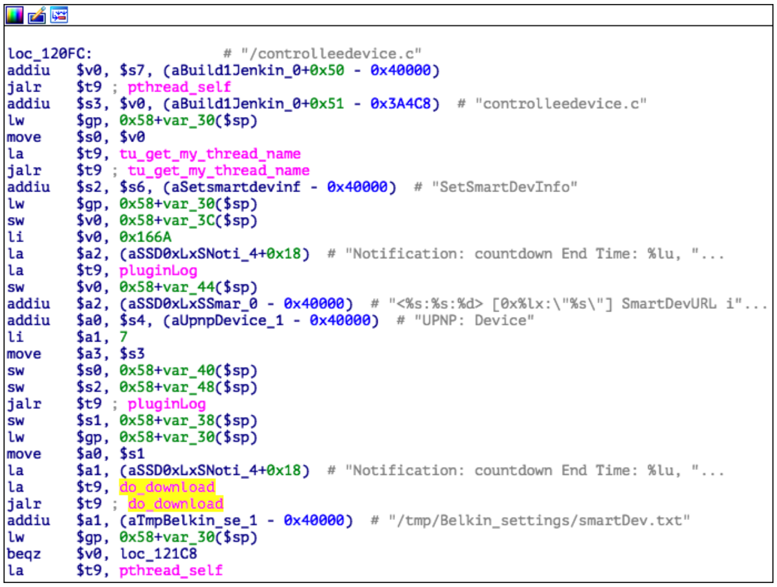


```
.text:004B6028                  move    $s5, $v0
.text:004B602C                  lw      $gp, 0x140+var_128($sp)
.text:004B6030                  addiu   $a2, $s4, (word_6EF02C - 0x6F0000)
.text:004B6034                  li      $a1, 0x6E0000
.text:004B6038                  la      $t9, websGetVar
.text:004B603C                  addiu   $a1, (aSyscmd - 0x6E0000)  # "syscmd"
.text:004B6040                  move    $a0, $s3
.text:004B6044                  jalr    $t9 ; websGetVar
.text:004B6048                  sw      $v0, 0x140+var_34($sp)
.text:004B604C                  lw      $gp, 0x140+var_128($sp)
.text:004B6050                  move    $a0, $s3
.text:004B6054                  la      $t9, websHeader
.text:004B6058                  nop
.text:004B605C                  jalr    $t9 ; websHeader
.text:004B6060                  move    $s7, $v0
.text:004B6064                  lw      $gp, 0x140+var_128($sp)
.text:004B6068                  move    $a0, $s0
.text:004B606C                  la      $t9, strcoll
.text:004B6070                  nop
.text:004B6074                  jalr    $t9 ; strcoll
.text:004B6078                  addiu   $a1, $s4, (word_6EF02C - 0x6F0000)
.text:004B607C                  lw      $gp, 0x140+var_128($sp)
.text:004B6080                  beqz    $v0, loc_4B610C
```

Figure 1. Decompile non-public API

We found another vulnerability in the form of server-side request forgery (SSRF) in WeMo SetSmartDevURL API. The API service usually opens a randomize TCP port from 49151 to 49155. If a user sends a request to set a smart device URL via this API, the service will try to fetch the URL with cURL library.

SetSmartDevURL – do_download (/lib/libUPnpHndlr.so)



```
loc_120FC:                 # "/controlleedevice.c"
addiu   $v0, $s7, (aBuild1Jenkin_0+0x50 - 0x40000)
jalr    $t9 ; pthread_self
addiu   $s3, $v0, (aBuild1Jenkin_0+0x51 - 0x3A4C8)  # "controlleedevice.c"
lw      $gp, 0x58+var_30($sp)
move    $s0, $v0
la      $t9, tu_get_my_thread_name
jalr    $t9 ; tu_get_my_thread_name
addiu   $s2, $s6, (aSetsmartdevinf - 0x40000)  # "SetSmartDevInfo"
lw      $gp, 0x58+var_30($sp)
sw      $v0, 0x58+var_3C($sp)
li      $v0, 0x166A
la      $a2, (aSSD0xLxSNoti_4+0x18)  # "Notification: countdown End Time: %lu, "...
la      $t9, pluginLog
sw      $v0, 0x58+var_44($sp)
addiu   $a2, (aSSD0xLxSSmar_0 - 0x40000)  # "<%s:%s:%d> [0x%lx:\"%s\"] SmartDevURL i"...
addiu   $a0, $s4, (aUpnpDevice_1 - 0x40000)  # "UPNP: Device"
li      $a1, 7
move    $a3, $s3
sw      $s0, 0x58+var_40($sp)
sw      $s2, 0x58+var_48($sp)
jalr    $t9 ; pluginLog
sw      $s1, 0x58+var_38($sp)
lw      $gp, 0x58+var_30($sp)
move    $a0, $s1
la      $a1, (aSSD0xLxSNoti_4+0x18)  # "Notification: countdown End Time: %lu, "...
la      $t9, do_download
jalr    $t9 ; do_download
addiu   $a1, (aTmpBelkin_se_1 - 0x40000)  # "/tmp/Belkin_settings/smartDev.txt"
lw      $gp, 0x58+var_30($sp)
beqz    $v0, loc_121C8
la      $t9, pthread_self
```

Figure 2. Decompile *do_download* function

An attack can use these two vulnerabilities together, which will remotely execute arbitrary code via WeMo API by one POST request.

2. Belkin WeMo® LED Lighting Starter Set – Unauthorized SYSEVENT Service

The Belkin WeMo® LED Lighting Starter Set contains a system daemon named *syseventd*. It is a system service from the RDK-B Utopia framework and could be abused to do remote code execution without authentication.

Sample command	Result
./sysevent --port 52367 --ip xxx.xxx.xxx.xxx async [hello] [/bin/echo] [everyone!] *./sysevent --port 52367 --ip xxx.xxx.xxx.xxx set [hello] [world]*	*/bin/echo hell world everyone!*

Table 2. Translated command for SYSEVENT

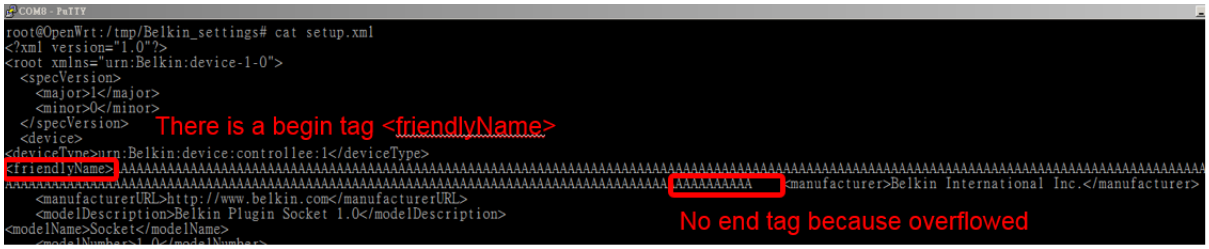
Because *syseventd* is running with root permission, an attack can seize complete control of the device after successful exploit.

References:
https://wiki.rdkcentral.com/display/RDKB/RDK-B+UTOPIA
https://github.com/rdkcmf/rdkb-Utopia/tree/master/source/sysevent

3. Belkin WeMo® Service – Denial of Service

Some Belkin devices have a common service running on them called WeMo, which is designed for home automation. While reviewing the service, we found a vulnerability in a WeMo API, which can change the friendly name for each device from the mobile app. The mobile app already limits the size of one's input, but an attacker can write commands directly to the device through the WeMo API with a few bytes more. These excess bytes cannot help an attacker to gain control of the device but instead will crash the device with proper modifications, prompting the WeMo service to crash immediately.

For example: An attacker can name the device a few bytes more, and the WeMo service will generate a device info on *setup.xml* and re-register the device with a new name from the XML file.



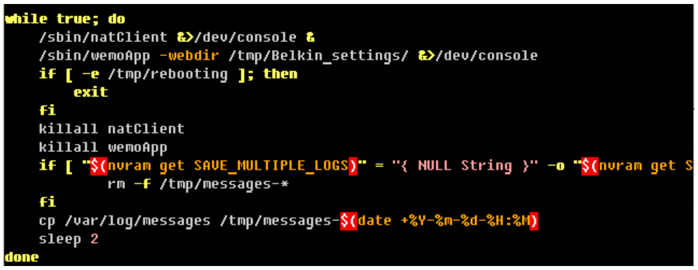
```
root@OpenWrt:/tmp/Belkin_settings# cat setup.xml
<?xml version="1.0"?>
<root xmlns="urn:Belkin:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
<deviceType>urn:Belkin:device:controllee:1</deviceType>
<friendlyName>AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    <manufacturer>Belkin International Inc.</manufacturer>
    <manufacturerURL>http://www.belkin.com</manufacturerURL>
    <modelDescription>Belkin Plugin Socket 1.0</modelDescription>
<modelName>Socket</modelName>
    <modelNumber>1.0</modelNumber>
```

There is a begin tag <friendlyName>

No end tag because overflowed

Figure 3. Broken XML file

The image above shows how an attacker controls the XML file to make the WeMo service crash. It is part of the WeMo startup script, which describes how the bug will continuously occur or be in a loop. The vulnerability will happen every time the WeMo service starts because *setup.xml* is damaged. */sbin/natClient* controls the Wi-Fi service of the device, while */sbin/wemoApp* controls the WeMo service itself. Since the loop takes a while before both services restart, and the wemoApp is likely to crash soon, the user will not be able to engage with the damaged device in time.



```
while true; do
    /sbin/natClient &>/dev/console &
    /sbin/wemoApp -webdir /tmp/Belkin_settings/ &>/dev/console
    if [ -e /tmp/rebooting ]; then
        exit
    fi
    killall natClient
    killall wemoApp
    if [ "$(nvram get SAVE_MULTIPLE_LOGS)" = "{ NULL String }" -o "$(nvram get S
            rm -f /tmp/messages-*
    fi
    cp /var/log/messages /tmp/messages-$(date +%Y-%m-%d-%H:%M)
    sleep 2
done
```

Figure 4. Bootstrap script for WeMo device

According to the device manual, there is a reset button on the device, which allows for a factory reset by pressing the button during boot time. However, the reaction of the reset button is too late, and the device remains on a crash loop.

The affected devices we verified were:

- Belkin WeMo® Switch Smart Plug
- Belkin WeMo® Insight Smart Plug
- Belkin WeMo® Link

- Belkin NetCam HD+ Wi-Fi Camera

We tested the denial-of-service vulnerability only on these devices, but we believe all WeMo devices have similar code.

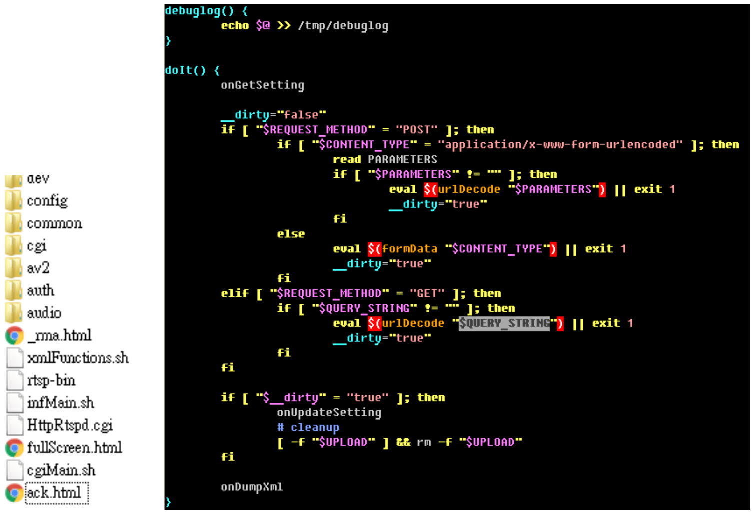
4. Buffalo WSR-300HP – Command Injection

The vulnerability, CVE-2014-8361 Realtek SDK Miniigd UPnP SOAP Command Injection, exists in the miniigd service provided by another manufacturer. The miniigd service provides mini UPnP service from UDP port 50000 to 60000. An attack can do remote code execution with proper modifications on existing exploit code.

Reference:
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-8361

5. D-Link DCS825L EyeOn Baby Monitor – Command Injection

We found that parts of the web framework are written in shell scripts. Additionally, upon reviewing the files, we found that part of the variables can be controlled from user input.



```
debuglog() {
        echo $@ >> /tmp/debuglog
}

doIt() {
        onGetSetting

        __dirty="false"
        if [ "$REQUEST_METHOD" = "POST" ]; then
                if [ "$CONTENT_TYPE" = "application/x-www-form-urlencoded" ]; then
                        read PARAMETERS
                        if [ "$PARAMETERS" != "" ]; then
                                eval $(urlDecode "$PARAMETERS") || exit 1
                                __dirty="true"
                        fi
                else
                        eval $(formData "$CONTENT_TYPE") || exit 1
                        __dirty="true"
                fi
        elif [ "$REQUEST_METHOD" = "GET" ]; then
                if [ "$QUERY_STRING" != "" ]; then
                        eval $(urlDecode "$QUERY_STRING") || exit 1
                        __dirty="true"
                fi
        fi

        if [ "$__dirty" = "true" ]; then
                onUpdateSetting
                # cleanup
                [ -f "$UPLOAD" ] && rm -f "$UPLOAD"
        fi

        onDumpXml
}
```

Figure 5. Vulnerable shell script

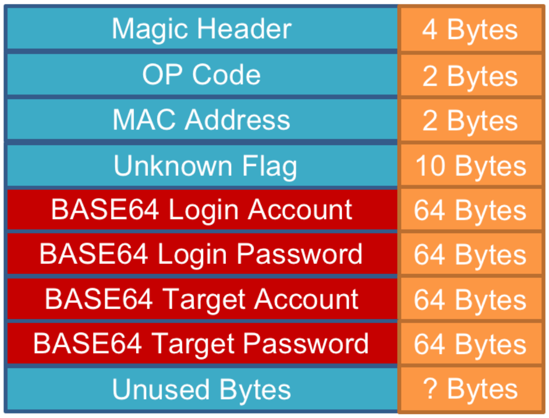
An attacker can forge malicious HTTP requests to execute commands on the device. Fortunately, the web server uses basic authentication first, before anyone can access any webpage.

6. D-Link DCS825L EyeOn Baby Monitor – Stack Overflow

A UDP "Discover" service, which provides multiple functions such as changing the passwords and getting basic information, was installed on the device.

An attacker can craft a malicious UDP request to perform stack overflow on the data by using proper ROP (return oriented programming) gadgets to execute an arbitrary code with root privilege on the device.

An example is the "Change Password" UDP packet format described below. Since the data was stored on stack, an attacker can use a large Base64 string on the credential fields in order to control the return address on the stack.



Magic Header	4 Bytes
OP Code	2 Bytes
MAC Address	2 Bytes
Unknown Flag	10 Bytes
BASE64 Login Account	64 Bytes
BASE64 Login Password	64 Bytes
BASE64 Target Account	64 Bytes
BASE64 Target Password	64 Bytes
Unused Bytes	? Bytes

Figure 6. "Change Password" UDP packet format

7. Dahua IP Camera and PTZ Camera – Predictable Recovery Password

Password recovery is an option used when users forget their passwords. As such, it is designed to be used in a convenient way. However, the implementation in the Dahua device we examined is not good enough in terms of security. We found a vulnerability in its recovery password generation algorithm. Although the algorithm needs several factors to generate the correct recovery password, the program skips one factor, which in turn will generate a weak password. Moreover, if the devices are using default factory settings, the rest of the factors will be easy to guess.

Generating a recovery password requires four factors, but since one factor is skipped in the code, only three factors are left to crack. Once those factors are acquired, one can log in to any target account with this backdoor password. The password can be used in the web interface or the API.

The three factors concerned are:

1. Target account name
   The attacker can choose “admin” in most cases because it is a fixed administrator account on the device.
2. Serial number
   The host name is the same as the serial number in default factory settings, so the serial number can be obtained from the greeting message shown during web or DVR login flow.
3. Hashed date string with customized algorithm
   The date depends on the time zone of a target device. It will be calculated by a customized algorithm, which will result in a short string (five characters).

Finally, an attack can calculate an uppercase of MD5 from a specific format, then extract the first eight characters as a recovery password.

TREND MICRO™

Securing Your Journey to the Cloud

Created by:

TrendLabs

Global Technical Support & R&D Center of TREND MICRO