The Sound of a Targeted Attack

Stephen Hilt
Trend Micro Forward-Looking Threat Research (FTR) Team
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.

Contents

4
Exposed Services

7
Debug Information

11
Sonos on the Internet

14
Information on Leakage

24
Sonos Control

30
A Case of Finding the Target

35
Phishing with Sonar

36
Exploitation

37
Hardware Hacking

45
Sonos Update

46
Conclusion
Internet-connected devices are growing in popularity every day. It's becoming more and more difficult to buy a device that isn't internet connected. Having devices connected to help user functionality, as well as size and cost, is a driver of why the popularity is growing. Despite knowing the risks in a lot of cases, we find ourselves looking at newer devices all the time, ones that add the functionality of having information in the cloud or use services that we already subscribe to.

For this research, we focused on internet-connected speakers. To look at the security implications of these devices, we wanted to find ones that are popular, have the ability to connect to the internet directly via Wi-Fi or Ethernet, and can pair with multiple devices such as duplicates of the device and mobile phones for remote access. This led us to the speaker systems of Sonos, a brand that has grown a strong reputation over the years due to good sound quality and a price that attracts a wide variety of people.

While the subject of this paper is the Sonos speaker because of availability and access to the device, the intent of the research is to focus on a larger issue that we are facing as internet-controlled sound devices are making their way into the home as well as businesses: Are there common issues across devices? From this stance, we also looked at other IoT speaker systems like the Bose SoundTouch Systems.

We have reached out to both Sonos and Bose regarding our findings. Sonos was quick to act and fix the issues regarding information leakage. The update they immediately released is discussed at the end of this study. We are also currently waiting for Bose to reply or come up with an update to address the gaps in security we found.
Exposed Services

The setup process for the Sonos speakers is straightforward and common to many IoT (Internet of Things) type devices. In this case, there are two methods for setup. You can plug an Ethernet cable into the back of the Sonos or you can connect the speaker over Wi-Fi. We set the devices up over Wi-Fi, and to do this the speaker creates an ad hoc access point that you connect to. And then follow the instructions within the mobile application on setup. If you chose the Ethernet method, your Sonos would be detected by the application and you skip the setup process over Wi-Fi.

Once connected to the network, we decided to see what ports were open on the Sonos to see if there were any basic issues, such as an FTP server or telnet running.

![Image of Nmap results for Sonos speaker](image1)

Figure 1. Nmap results for Sonos speaker

Unsure what each of these ports are, we ran nmap with the version probes (-sv) and ran the default nmap scripts (-A) to see if we could gather some more information about each of the ports and services that were running on the Sonos.

![Image of Services shown from nmap scan](image2)

Figure 2. Services shown from nmap scan
While checking to see which ports and services were open, we found that only three ports were open on the Sonos itself. These were two Universal Plug and Play (UPnP) services and one encrypted UPnP service. Launching the Sonos application while running Wireshark showed that over SSDP (Simple Service Discovery Protocol), the applications were pointed to communicate with http://<ip>:1400/xml/device_description.xml. Following some communications, it also appeared most of the communications were over TCP/1400 as the SSDP message below shows.

![SSDP packet captured in Wireshark](image)

Based on this information and other work and postings on the Sonos, it became obvious that port TCP/1400 was the port we wanted to focus on first. In fact, a developer named Barry Caruth had already written an Nmap script to parse some information from this port. Figure 3 shows the results from that Nmap script showing basic information about the Sonos device.
When looking into the script itself, it looked like the information was coming from a specific URI path of `/status` on the device (http://xxx.xxx.xxx.xxx:1400/status).
Debug Information

When browsing to the local Sonos system on the URI that was provided in the Nmap script, a lot of subpages were shown via a simple website. This site contained things that are obviously being utilized to populate the applications that control the Sonos system as well as some items that are used for the initial setup.

Some of the interesting subpages have content that Sonos probably never thought would be exposed directly to the internet. This site, with no authentication, allows you to see information about the tracks currently being played, what music libraries it knows about, what devices have ever connected to it to control it, and down to personal information such as emails associated with specific audio streaming services like Spotify.
Sonos has also allowed many debug functions to be exposed via this website. This includes the ability to traceroute, ping, and even make an mDNS announcement via a simple website.

![Figure 6. tools.htm page within the Sonos system](image)

Using the ping utility, you can easily find out the list of other active devices on the network by pinging the devices individually (given that you have collected the internal IP address of the Sonos device from the `ifconfig` page within the `/status` menus), or you can see what also responds via a broadcast ping.

![Figure 7. Example of a broadcast ping](image)
We looked into these pages to see if there were any vulnerabilities that could be exploited to take over the Sonos Play:1 test unit. Sonos had some good designs, in at least the firmware versions we were testing on, that have input validations as well as a CSRF-token to prevent some attacks. For example, if you type 127.0.0.1;ls in the ping field, you would get a response of Error 400 Bad Request.

![Error 400](image)

Figure 8. Error shown if command injection is attempted

Ultimately, if attackers wanted to see and collect all the hosts and what types of hosts they may be after doing various discovery methods via the Sonos system itself, they would then browse to /status/proc/net/arp page and collect more information about the devices on the network that the Sonos has been able to contact.

![ARP table from the Sonos device](image)

Figure 9. ARP table from the Sonos device

Let’s say an attacker knows the target uses a Sonos device. The attacker can then take the information collected here to tailor better attacks against the target. This could include mobile devices, printers, and even types of computers on the networks.
Within the status page, there is also a page that is titled “scan results.” This runs a command that looks for wireless access points nearby. This would be utilized during the setup process to determine which wireless network you would want to connect the Sonos system to, and we will detail more about the risks of this sort of information in the information leakage section of this paper. First, however, let us examine the amount of exposed Sonos systems on the internet.

According to W3.org, the 400 Bad Request states, “The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.” Meaning that adding the semicolon caused an issue and the request could no longer be understood. This is likely due to some back-end argument checking. To further check through the webpages, we used Burp Suite to look for any other issues within these pages that may be exploitable. However, we were not able to find any vulnerabilities from an external web interface within the pages listed on /status.

With this information we set out to see how many of these devices were exposed on the internet.
Sonos on the Internet

We worked with Shodan to scan the internet for Sonos devices based on the Nmap script that we previously discussed. To query Sonos devices within Shodan, you will want to use the query parameter “Sonos port:1400.”

![Sonos devices in Shodan](image)

The number of Sonos devices have changed from around 4,000 shown here to up to around 5,000 devices during the course of this research. A lot of this is due to the fact that the IPs that these are on are not fixed IP address ranges; often they are on residential lines. But what remains the same is that the United States and Sweden are the top two places where Sonos speakers are exposed on the internet.
What is surprising about this is that new products that have been released recently are showing up online. The PLAYBASE is a new product that allows you to connect your TV's audio output to a Sonos system. This product was released early 2017.

Although the Bose SoundTouch line has fewer units that have built-in Ethernet connectivity, this doesn’t prevent them from winding up exposed online. Much like the Sonos system, the SoundTouch has an unauthenticated API that allows full control over the speakers. A lot of work has also already gone into making libraries that work with these speakers. In fact, there is also a Python library.

Based on this, we can determine that on port 8090 there is a web service running and that /info has some of the basic information that is needed to connect to the device. Based on this information, we did some digging in Shodan to see if we could locate any online.
Looking first into port 8090 and keyword Bose got us to a single host. This host had a server type of Bose-Software. We then searched based on that to find anything else without the port number and this led us to a lot of UPnP devices that had the server type of Server: Allegro-Software-RomPager. Partnering this with the port number of 8090, we searched for what we believe to be Bose SoundTouch devices. While there were fewer than the number of Sonos devices, the numbers are significant and likely to grow as these devices are being directly exposed to the internet.

The Bose SoundTouch 10 does not have an Ethernet port, so these devices are being exposed directly to the internet via a misconfigured wireless router, or via UPnP which is being allowed to open to any host to connect to it.
Information Leakage

Within the Sonos control website are pages that have a plethora of information depending on how things are configured. One of the first pages we explored was a page labeled “topology” from the status page. This page keeps up with what devices have connected to the Sonos system to control it, and what other speakers this Sonos knows about. In Figure 15 we have a list of mobile phones that have connected to the Sonos system.

The Name field is the name configured on the device. Here we see that one of the phones was called APTPhone. This is the name that is under the about section of the iPhone in settings. The Location mentioned is the IP address of the remote device. This could be used to help gain some reconnaissance information regarding devices within the same network as the Sonos unit, and where they may be inside the network segments.

Next, we have the UUID, which, in this case, is what type of device it is, plus a random string afterwards that is not tied to a serial number or anything on the device. In fact, it looks like this changes every so often as we have five devices that have the same name. Given that the devices that were connected to it weren’t altered, this led us to believe that these are changing because of something else within the application that was downloaded from the app store. Other names we saw were Android phones that controlled the devices; names like “ASUS Transformer Pad TF300T” and “Samsung SGH-T599” and other variations of Samsung and other vendor phones like Motorola, HTC, and so on. What this gives an attacker is some ability to find out what devices are in the networks or that a person may specifically have.

Figure 15. Topology page showing what devices have connected to the Sonos
Not only were some of the devices we observed local to the Sonos device (meaning within RFC 1918 space), but we also saw addresses that were on the same internet routable IP space as the device. This could be because the device was plugged straight into the network, or because someone had used NAT to route the IP address through to the internet, allowing anyone to connect to the Sonos, and then the application on the controlling device used TCP/1400 to connect to the devices’ SOAP XML interface to control the devices.

While this provides a good deal of information about the users, we can also find out more about the person if we go and take a look within the accounts page of the Sonos status page. This page keeps track of the accounts that are tied to the Sonos system. We tried a few music services such as Spotify, iHeart Radio, and Pandora, to name a few. This page includes a Serial Number that is just a basic number to keep track of the number of accounts, and a Type, which appears to denote the type of account (in our case, 3079 was Spotify and 3 was Pandora). The UN is the username for the account. Pandora was the only one that had an email address shown in our tests while the rest of the accounts used some kind of token that was generated. Next, NN is the Name applied to the account when configuring the device. This is a user-defined field and can be edited at any time. It defaulted to the first name of the person on the account; in the case of our test unit, it defaulted to ‘stephen’ for all the accounts. The OADevID looks like an ID used by the Sonos systems to communicate with the services. And then finally a Hash, in which only Spotify had a Hash. Initially, this Hash looked like it could have been base64 but it is not. And based on some additional looking, it does change with the length of the password for the Spotify account.

Based on information on these pages, we coded a small script that would query the publicly available Sonos systems on the internet, read the accounts page, and parse out these fields. When we finished the parsing, we had 1,293 emails of which 727 were unique. Email addresses were then loaded into Maltego, an open source intelligence (OSINT) tool, and we ran multiple transforms on them. One transform was one that queries the website https://haveibeenpwned.com/, which keeps track of whether your email address has been seen in any breaches.

The Maltego graph got a little large given the size of the breach data and the number of email addresses. Showing the most common breached sites was a little cumbersome to show in the full Maltego graph as the one in Figure 17 is zoomed out to 10% to have all the nodes fit.
When you zoom in, you can see some of the top breaches listed, the ones we all know and likely had to deal with by changing passwords, or other issues that may have been caused by these breaches.

The most common breach was the LinkedIn breach, with 177 accounts. This breach also had the password hashes released and people around the globe have cracked these passwords to use with penetration tests, or as malicious actors trying to break into accounts. Knowing these breaches and also what sites they came from could then lead an attacker to log into the users’ accounts identified from the Sonos. Attackers can also check if the user hasn’t changed passwords after the breach – an all too common occurrence.
Much like the Sonos, the Bose SoundTouch also has a page that keeps track of the accounts that are linked to the device. Here we show the /sources page from the SoundTouch that shows the Spotify username which is the source account, as well as the email address that is linked to this Spotify account. Using the same methods from above, we could perform the same steps against any email addresses that were found within the Bose SoundTouch sources page.
This is not the only page that an attacker may want to scrape to see information from the Sonos device. To aid in finding what is near the Sonos speaker, the speaker scans for nearby wireless access points (AP). This is done during the setup process to have the Sonos know what Wi-Fi AP you want to connect to. During setup, you’ll choose the one you want to connect to, and what the password for the connection is. This is done over an SSID that the Sonos starts and you connect to. Once the speaker is associated to a wireless access point, you don’t see this option unless you want to change what Wi-Fi it will connect to. Based on testing, if you are using the Ethernet port on the Sonos, this does not happen.

While trying to figure out how to use this information to identify the owner of the Sonos system, we came across a website that compounds multiple sources of Wi-Fi geolocation. The website of Alexander Mylnikov has an API that you can use to query specific BSSIDs and find the latitude, longitude, and the approximate range in meters of where the access point is.

Based on the BRCM_TEST_SSID BSSID above, the API shows the latitude, longitude, and the range of 133.2 meters for the information pertaining to the test device.
This information can then be plotted on Google Maps to get a better picture of where the BSSID may be seen in the world. According to the API results, this particular BSSID is located in Zurich, Switzerland.

Figure 23. Approximate location of BRCM_TEST_SSID

To expand on this information, we once more wrote a script to scan publicly accessible Sonos systems on the internet. This time, we tried looking for BSSID and SSID details, then looked them up in the mentioned API for location, and stored them as a KML file to load into Google Earth. There were multiple things we wanted to see while trying to do this. For example, we wanted to see how accurate the grouping of the SSIDs were in a given area to figure out approximate locations of the Sonos systems.

At the time of our survey of exposed Sonos units online, we found 12,311 BSSIDs. Out of these, 10,219 had geolocations inside the API. A quick scroll around in Google Earth showed that there was some pretty good accuracy in the location of these access points. Here is an example showing the location of SSIDs in a location in the Netherlands. Based on the groupings of these SSIDs, you can get a general location down to a few blocks where the Sonos speaker is likely located. While completely possible, we chose not to narrow down the locations of these devices even further for this case study.
Figure 24. High-level view of parts of Europe with exposed Sonos devices

Figure 25. Zoomed-in area in The Netherlands
These are just a few of the pages wherein we examined information. There are plenty of other types of information on a few other pages that could be utilized. For example, in the “wpa_supplicant.log” page, you can see what access point the Sonos is joining, and partner this with the already-collected information to get a closer approximation of the location of the Sonos speaker. Sifting through the collected SSIDs, we were able to see the AP names as well.

<table>
<thead>
<tr>
<th>^_^</th>
<th>Princess Leia’s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hogwarts Great Hall WiFi</td>
<td>This is Wifiiiii</td>
</tr>
<tr>
<td>NoFreeWifiForYou</td>
<td>Back office</td>
</tr>
<tr>
<td>Pretty fly for a Wifi</td>
<td>Nacho WiFi</td>
</tr>
<tr>
<td>A_LAN_DOWN_BY_THE_RIVER</td>
<td>Bill_Wi_the_science_Fi</td>
</tr>
<tr>
<td>Ermahgerd We Fer</td>
<td>404 Server Error</td>
</tr>
<tr>
<td>Router? I hardly know her</td>
<td>2ez4me</td>
</tr>
</tbody>
</table>

Table 1. Some of the Wi-Fi SSIDs we found

The Bose SoundTouch had a more limited view, based on what we have seen that from the port 8080 on the device, there is a configuration page that shows you the neighboring SSIDs and allows you to connect to one. While this doesn’t present you the same information such as the BSSID or RSSI that the Sonos does, using services like Wigle, it is possible to look up approximate locations based on the SSID itself.

Figure 26. SoundTouch Wi-Fi configuration page
Using a Wigle account, it is possible to search the SSID and find a location based off of the SSID. If there are multiple SSIDs, you can partner this information with the IP address geolocation to figure out a more accurate location for the SSID. Figure 27 is an example of searching for the SSID shown in Figure 26 in Wigle. This shows that the Bose SoundTouch device is in the Baltimore area. We can perform with the same results as we did with the Sonos with multiple SSIDs.

Figure 27. Location of SSID according to Wigle

Another cause for concern is the shares page. This page shows any mounted shared drives that are used in the library. We tested this and it provided a list of all songs that were on the shared drive that were playable by the Sonos system. This also includes any local folders you have shared with your Sonos speaker. When setting up the Sonos you have to manually add these — they are not auto-discovered.

Figure 28. Example of the Sonos Add Music folder page
Once you have added any of the above options, they will show up in the shares page on the Sonos status page. This information could be utilized to collect more information about a potential target of an attack. If an attacker finds out what type of music or even an artist the user liked, it may provide an avenue for an attack. For example, the attacker could craft a spear-phishing email leveraging social engineering, or promising tickets to an upcoming gig of the target’s favorite artist.

Knowing that these speakers are very popular, a possible attack scenario is that an attacker could write a web application that scans for this information via a client-side attack. This could be done within a Java Applet, or if there is a vulnerability within the browser, by using a CSRF (Cross Site Request Forgery) to try to scan a local subnet via the browser. This could gather information about the Sonos that are not directly exposed on the internet and gain the same information as shown previously. Also, malware could be utilized for the same attack trying to collect more information about these devices and their owners.
Sonos Control

While there is a lot of information that is on the status page of the Sonos, there is still more information we can learn by interfacing directly with the SOAP XML interface. When looking into the information that is sent back and forth between the client and the Sonos system, you find that you can see in plain text the commands that are being used to control the Sonos, such as pause, play, skip to the next track, and change the volume. Of course anything that you can do with the app you can do over this SOAP XML interface also, as we will show it has no authentication or encryption. When you watch the commands in a tool such as Wireshark, you can see all this information. Below is an example of the “Next” command within Wireshark.

![Figure 29. Sonos Next command sent via user interface](image)

We wanted to see if there is anything within this packet that could hinder us from resending the same data again and be able to skip songs. After further inspection of the packet, it was determined that a packet replay would likely successfully change the song on the Sonos system. To test this, we took the packet from Wireshark by selecting it and copying the Hex Stream. We then stripped the Ethernet and IPv4 headers off the packet and sent the information with the code below. As expected, the song went to the next track. Figures 30 and 31 show the script running, and the Sonos application running to alert us that the next track is playing and what it is. The Sonos application does not need to be running to change to the next track — This is only included to visually show the change in this paper.
25 | The Sound of a Targeted Attack

Figure 30. Python code to change to the next song

```python
import socket
import sys

# Argument handling
if (len(sys.argv) == 2):
    print("USAGE: Python ./NextTrack.py <host>")
    sys.exit()

# Host is passed in via CLI arguments
host, sys.argv[1]

# Sonos port
port = 14000

# convert into python hex format (x)
hex_string = "504f5345474c4543484c4c00"
hex_data = hex_string.decode("hex")

# open socket
try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))
    print(\"Connection successful.\")
    sock.send(hex_data)
except socket.error:
    print(\"Connection unsuccessful.\")
sys.exit()

# Close socket
sock.close()
```

Figure 31. Sonos notification saying the next song is playing
Before going much further in looking into what else we could code to control the device, a simple search showed us that there is a preexisting Python Library called SoCo that was written to help control the devices. There is even an example web application to control your Sonos devices. It’s key to point out that there is no authentication for any of the methods on the Sonos SOAP XML interface. This means that anyone can control your device or watch the tracks, or even just monitor what songs you are playing. This could be done to learn more about the person who owns the Sonos, or it could be done to determine when someone is around the Sonos and is listening to music. In the case of attackers, given they already know approximately where the devices are located, they can now use this to determine when a person is physically at home or not by observing these patterns over time. A burglar could also use this knowledge to determine when the property is vacant and go into the house based on these patterns. To test this idea that we could monitor what was being watched, we decided to write up something using SoCo that pulls the currently played track.

```
import sys
from sooco import SoCo

# Function to get current track info:
if len(sys.argv) == 2:
  print("USAGE: Python ./CurrentTrack.py <host>"
  sys.exit()  

sonos = SoCo(sys.argv[1])
track = sonos.get_current_track_info()
artist = track["artist"].strip()
title = track["title"].strip()
album = track["album"].strip()
length = track["duration"].strip()
time = track["position"].strip()

print("Artist: %s | Title: %s | Album: %s | Length: %s | Current Time: %s"
(artis_t, title, album, length, time))
```

Figure 32. Python code using SoCo to list details about the current track

We found that in the response, there is a field that shows how long the song is, and what the current time in the track it is at. This could be used by an attacker to determine if the song is being played or paused as well as to help determine if the person is actively listening to the track, or if it is paused based on the time between queries. Shown here is an example of the above script running on our Sonos test unit. This shows that the Sonos is currently being played.

```
$ python CurrentTrack.py
Artist: Rick Astley
Title: Never Gonna Give You Up
Album: Whenever You Need Somebody
Length: 0:03:32
Current Time: 0:00:11
$ python CurrentTrack.py
Artist: Rick Astley
Title: Never Gonna Give You Up
Album: Whenever You Need Somebody
Length: 0:03:32
Current Time: 0:01:31
```

Figure 33. Sonos current track current time example
Under the SoCo project, there is an example of playing a song by providing it a URI path. Upon further searching, we found that there is a project called Ghosty that utilizes the Ruby scripting language to play songs on a Sonos at random times that mimic a ghost or a creaking door. While this sounds like a simple piece of fun code, there are real-world examples in the Sonos community forums of this happening to people, and not just creaking doors and ghost sounds. One post described the user’s experience of hearing sounds similar to “an explosion from a movie.” It is likely a scary encounter to have your speaker turn on and make an explosion sound that repeats until you unplug the device itself. One user responded to the original poster and said they should check out the diagnosis page that is located at status/perf. This information can be used for any forensics you may have to do on these devices as it keeps a record of what IPs are connecting to the Sonos device and what commands they are running. Here is an example of our perf page.

![Figure 34. Sonos status/perf page](image)

After the command such as `next()`, there is an IP address followed by the OS and also the type of computer at the end of the entry.

In recent news, there was an episode of South Park that made people’s Amazon Echo units add things to their Amazon wish list. When we saw the episode, we were amazed because we had tried the exact same thing for this research, but the episode aired before we could publish this. Congratulations Eric Cartman for beating us to this research.
Based on the SoCo example to play a URI path, we coded up an example where we hosted the sounds of Alexa commands as a proof of concept. For the audio content, there were a few options. We could simply record our voice, something that we have done a few times before in earlier research where we showed that you can control an Echo with a human voice. Instead, we chose to use the say command within Linux and created an MP3 and hosted it on a local web server. Once we had the sound file, we ran our script to play a URI path.

Figure 35. Python script to play URI path

Once played, the Echo performed the task that we tested. We tested a few commands, which are listed in the table below along with the outcome.

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Alexa what time is it”</td>
<td>“The time is 3:12 pm”</td>
</tr>
<tr>
<td>“Alexa rap for me ”</td>
<td>Connect sync link all the pieces of your life I Get it done at the speed of WiFi I’m the player, the coach, the arena, the game If you want something done, you just gotta say my name</td>
</tr>
<tr>
<td>“Alexa play song”</td>
<td>“I can’t find any music in your library”</td>
</tr>
<tr>
<td>“Alexa What’s on my shopping list”</td>
<td>“Our shopping list is empty”</td>
</tr>
<tr>
<td>“Alexa &lt;item&gt; to shopping list”</td>
<td>“I have added &lt;item&gt; to your shopping list”</td>
</tr>
<tr>
<td>“Alexa turn off hue light”</td>
<td>Lights turns off “OK”</td>
</tr>
</tbody>
</table>

Table 2. Commands sent via Sonos to Alexa

With the Bose SoundTouch devices, you can also play a URL. Inside of the libsoundtouch documentation, it mentions that you can point it to a URL just as with the Sonos systems. This was not tested as we did not have a Bose SoundTouch test unit to issue commands at. However, this allows the same commands to be issued as we have demonstrated above.
As with the Sonos, it is also possible to get the current track playing on the SoundTouch devices, and this information could be used in the same way as it can be with the Sonos. However, one noticeable difference is that the source in which the track is being played can be gathered easily. Below is the example of a device that was queried.

```python
# Play URL
device.play_url('http://fqdn/file.mp3')
```

Figure 36. play_url function in libsoundtouch

Figure 37. Song currently playing on the Bose SoundTouch
A Case of Finding the Target

Combining everything we now know about how information was being leaked, and having determined more information about the current state of the Sonos speaker, we decided to do a case study of one user to determine if we could find out more about the person, and the area where they live. We chose a Sonos speaker at random and proceeded. The identity of the person won’t be revealed but the steps we took will be outlined.

In Colorado, there were a couple of Sonos devices shown in Shodan. In total, we found six exposed devices online at the time of testing and chose one at random.

Once we had the device IP, we browsed to the accounts page, viewed the scan results page, and looked up the BSSIDs as we had done before. This led us to go down classic open source intelligence routes to find more information about the person.
First, the accounts page gave us a Pandora account. We know it’s a Pandora account due to the Type being listed as 3, which is the same as the Pandora account that was linked to our Sonos test unit. This email address was recorded and put into Maltego, where we ran a few transforms on the email address and found only one really useful bit of information — which is that the person has a Pandora account with that email address (no surprises there).

Viewing the “scanresults” page provided us with the general location of this Sonos based on the BSSID geolocation lookups via the previously mentioned API.

![Figure 39. Approximate location of the randomly chosen Sonos unit](image)

Based on this information, we started trying to figure out the exact location of the Sonos based on the other information that is in the Sonos website.

For the email address, we cross-referenced the subject user’s email address with Pipl, an online search tool used to locate people based off of public information. In doing so, it was possible to find a name associated with the email address.
Sites like Pipl should be taken with a grain of salt, so we went on to verify the information by searching for the email address on Facebook to see if we could generate the same account by email address. Using the site IntelTechniques.com, you can search Facebook for an email address. In this case, the result on Facebook was that we could not tie the email address to an account, as shown below.

This is fine as this could be because of a privacy setting that doesn’t allow people to search for the user by email address. So, another way to search for an email address is to click the “Forgot account” link and use the email address as the input.
In this case, we can see that this is a valid email address for someone on Facebook, but we aren’t presented with a Facebook profile picture or anything else that may help us tie the two accounts together based on the Pipl search. At this point, we normally wouldn’t consider these accounts verified. But knowing the neighboring access points and a possible name of the person who owns that email address, we can attempt to find an address for this person in the town.

Using a website called FamilyTreeNow.com, or any other site that has public records on it, we can start to see if we can find a possible address for this person, and we found a few potential addresses. The state this user was in also has a really interesting public database that gives the current address of a person and their voter information. This information, along with the information from FamilyTreeNow.com, led us to an address that was highly likely to be correct. In normal OSINT, there is no really great way to determine if this is the address of the person, other than based on records.
Since we already had local scanning of nearby SSIDs and our previously discussed method of mapping these out, we were able to plot the address from our OSINT investigation and see if we were close to those access points. As can be seen in Figure 44, the suspected current address above mapped to a location that is directly beside the previously shown SSIDs.

![Figure 44. Address mapped with SSIDs](image)

Now we had to figure out if we could find what type of music the person may be into. Running our previous script that pulls the current track revealed that the person may like 80s hair metal, or is into classic rock based on the selection that was currently playing.

```
$ python CurrentTrack.py
Artist: Extreme
Title: He-Man Woman Hater
Album: Extreme II: Pornograffitti
Length: 0:06:19
Current Time: 0:00:00
```

![Figure 45. Song currently playing on Sonos](image)
Phishing With Sonar

With the ability to target users down to knowing the exact house they are in and then confirming it based on only the information that’s within the Sonos and OSINT, an attacker could take the information and send them a targeted phishing attack to try and get them to become a victim of malware. In this attack case, however, we can now add sound to make the phish more believable – much like when people are fishing with fish finders that use sonar, a system that emits sound pulses and listens for them to bounce off an object. In this case, an attacker can use sound as well. However, it won’t be a pulse of sounds but a directed message to partner with the email. Needless to say, we did not go so far as to use this attack scenario on the individual in our research. But we did verify everything on our own test setup.

Let’s assume that an attacker knows your model number, serial number, and all the other information about your Sonos. They can then take that information and craft a message that states “Sonos System Alert: Your Sonos Play:1 speaker is having some technical difficulties, instructions on how to correct this can be located within an email sent to your email address that you have registered with.”

Partnering the email with the audio alert would make the message even more believable to even the most skeptical end user, such that the likelihood of getting the person to fall for the phish would likely be more than if this was just an email sent to the email address that was found within the Sonos system.
Exploitation

During the research, we also wanted to see if we could find any exploits that may be remotely leveraged by fuzzing some of the SOAP XML interface fields. This started with us writing a custom script to first fuzz the UUID field. While playing a song on the Sonos speaker, we ran our script and quickly made the Sonos stop playing songs and remove itself from the application to control the Sonos. As shown here, it sends an IGMPv3 leave group message as well as an MDNS query to flush the cache for the Sonos name. While this is not a buffer overflow or anything at this point in time, there are some mishandling issues if the values are non-ASCII values. In this case, you see the UUID is shown as 0x352494e434f4e5r3934394633453133364133303134303a0d. All the UUIDS end with 0x0a0d as that’s a new line and a carriage return value in hex. If the values have an ASCII to start with such as AAAAAA_AAAAAAAAAAAAAAAAA00 and this proved to not crash the system, the music continues to play. So, it appears that parsing non-ASCII values is causing this issue.

![Figure 46. Non-ASCII characters causing the DoS](image)

Because we stopped the tracks at will from playing and we disappeared from the application, an attacker could also do this method to have their phishing attack be more successful. If the attacker sent an email from product support and said that it had noticed the Sonos system is having an issue and to click a link to learn more information and get support on how to fix the Sonos system, again the likelihood of success would go up.
Hardware Hacking

With IoT devices, there is always the question of how the hardware is implemented. With the spread of IoT botnets like Mirai, we looked closer at the exposed systems. We felt like there were enough of these online that if there was an issue found with the firmware, there was a large enough user base for an attacker to create a botnet. Not only can an attacker script a ping-based DDoS bot already (based on the page shown earlier), but we wondered what else could be done if there is access to the hardware?

First we checked what was possible to do with the firmware file. We attempted to capture the files over a network pcap, then extract the firmware from the pcap. This turned out to be a waste of time because there is an “upgrade.log” file on the status page of the Sonos. This page contains a URL string that when you copy and paste into a browser or cURL/wget you will get the firmware file that the Sonos is running. It turned out that, based on the two models we had, there is a PowerPC version and an ARM version of the hardware. We focused on the ARM version because it seemed to be the most prevalent and had the most number of tool sets that could work with the firmware files if extracted.

We did some looking into this and there were examples of people on forums that had downloaded older versions of the firmware and were able to extract it and read the contents. However, this was not the case for our firmware files that we downloaded. Most of them were gzipped firmware with custom entry points.

We attempted to find older versions of firmware to see if there were any versions we could locate that were able to be unpacked. We found an older version of 29.5-90191-1-5. This version was able to be extracted via binwalk where we had a few files that were extracted.
The files vmlinux.29.5-90191.bin and 8EC90.cramfs were extracted, then using the uncramfs command from the firmware-mod-kit found on github, we deflated the cramfs to reveal a Linux folder structure.

In each of these folders, such as etc, there are typical files such as shadow and passwd. However, these files have no contents. We tried multiple times with multiple tool sets and were never successful in getting an older version unpacked. Based on information online, it looked like we would have had to go back to a version of firmware before 20.0 to be able to get any information from the extracted files anyway. Because of this, we moved on and disassembled the Sonos test unit.

A good resource for IoT devices is ifixit.com, where they have a manual with pictures on how to take apart the Sonos Play:1. Once we had taken it apart based on the instructions on ifixit, we started to look at any areas that we may want to spend some time on to determine if we can locate anything that would lead us to either console access to the OS or a method to extract the firmware.
First thing we started looking at were the chips that were in use. Here is the layout of the front and back of the motherboard in the Sonos (ARM version). Once you remove the RF protection cages around the equipment, you’ll see the processor, memory, and a bunch of other components. One interesting note is that as hardware breakdowns go, this one was easy but interesting as things are hot glued and there is foam around areas to prevent rattling inside the speaker.

![Sonos motherboard](image)

Figure 50. Sonos motherboard

The chips that we dug into were the Freescale CPU and the Winbond chip shown in Figure 51. The Winbond chip in the image on the right is the NAND Flash that is used by the Sonos for storage. On the left is the Freescale ARM processor that is a dual-core ARM chip.

![Closeup of the Winbond chip](image)

Figure 51. Closeup of the Winbond chip
Before looking into the chips much further, we looked into whether there was any low-hanging fruit on this board, such as an open serial interface, or a JTAG interface for us to tap and interface with. There were a few different locations that we tried initially to see what kind of data there may be on the board. Next to the Ethernet port were a set of pads that resembled a possible location of a UART port, or another type of interface. However, when we probed this, there were no signals on any of these pads and they appeared to be disabled.
Below the wireless card, there is another interface that we probed. This interface had data on it and it looked to be a Serial Wire Debug (SWD).

Once we had attached some pins to the pads, we were able to connect our Saleae Logic Analyzer to these ports and attempt to see if we could find any information on it. Allowing the Saleae to auto decode it as SWD and playing with the configuration of which of the pins was what input (SWDIO and SWCLK), the best we could find is shown below, where it appeared to have some SWD on the interface.
Using an STM32 F4 Discovery and attaching the pins to the corresponding pins on the board, we attempted to see if we could interface directly via SWD. However, we never were able to connect to the device via SWD despite our best efforts. Below is a screenshot showing the st-info command showing that once you remove the CN3 jumpers from the board, this turns the board to Discovery mode. This means that any device attached should show up with a proper serial number much like the device did with the jumpers on. This was never the case no matter what we attempted, so we moved on and assumed this is not SWD anymore.

Next, we found what could possibly be a JTAG interface next to the cage where the ARM chip is.
Under the wireless card itself are eight other pins that are in a staggered placement. These pins, much like the JTAG interface, are labeled in order.

Next, we found where any of the possible traces coming out of the Freescale processor may have vias that we could probe, and used the Saleae to determine if there were any signals present. This proved to find nothing of value for anything that we knew what to do with. Most of them at the time of probing had no signals on them.
As with any research, there are successes and failures. In this case, trying to find any interface to directly interact with the Sonos systems via hardware was not a success. While we focused on the Sonos Play:1, some older devices had an exposed UART that was pretty clear; as an example, the Sonos Play:5 is this way according to some of the forum posts that we saw on the internet. This could help provide us with some information that were missing on the Sonos Play:1. These looked to be of the PowerPC variant and not the ARM, but the underlying functionality is almost the same minus some of the features we were using to target some people such as wireless scan results and review log files.

Special thanks to Kevin Finisterre for the hardware help.
Sonos Update

We reported our findings to Sonos before this paper was published. Sonos was quick to respond and started remediating some of the issues. Sonos has fixed the DoS problem and now returns with an HTTP error code 412 (Precondition failed). In addition, now you can no longer see the scanresults page or many of the other pages that provide some information leakage. There are a few left such as the accounts page. Figure 61 shows the Sonos status page after the update.

![Updated Sonos status page](image)

Figure 61. Updated Sonos status page

While the update is a step in the right direction, you can still control these and gather more information from the devices with what is still being exposed via the status page. This is as of the software date “2017-09-27 16:26:54.868089” and software version number 37.12-45270. If you have not updated to this version of the software, you should update now to limit the personal data that is being exposed via these services, even if you are not directly exposed to the internet.
Conclusion

With the popularity of IoT devices growing every day, it is very important to be knowledgeable of the built-in security of these devices that ultimately could affect the owner and make them a target of an attack. While this research focused on Sonos speakers, we do not at all want to single them out as the only IoT device with security issues on the market today. As we showed, Bose SoundTouch also shared a lot of problems with the Sonos according to our findings. This is a common issue we observe in IoT devices, where there is a lack of security built into the design. With the Sonos, this was the case with the unauthenticated SOAP XML interface, and the web interface that leads to the information leakage. While these devices are never supposed to be exposed on the internet, we have shown that they can and will find their way directly on the internet. We believe that the manufacturers should do whatever they can to make sure that their devices are secured enough that if it is placed on the internet, the likelihood of attack is really low.
TREND MICRO™
Trend Micro Incorporated, a global cloud security leader, creates a world safe for exchanging digital information with its Internet content security and threat management solutions for businesses and consumers. A pioneer in server security with over 20 years experience, we deliver top-ranked client, server, and cloud-based security that fits our customers’ and partners’ needs; stops new threats faster; and protects data in physical, virtualized, and cloud environments. Powered by the Trend Micro™ Smart Protection Network™ infrastructure, our industry-leading cloud-computing security technology, products and services stop threats where they emerge, on the Internet, and are supported by 1,000+ threat intelligence experts around the globe. For additional information, visit www.trendmicro.com.