Anatomy of an IoT malware attack

How to prevent your IoT devices from joining the zombie bot horde

J Steven Perry

October 31, 2017

If you have IoT devices in your home, or on your corporate network (or both), they are under attack. They might have already been attacked and compromised. And you might not even know. In this article, I will address these three questions: What does an IoT device look like under the hood, what does an IoT malware attack look like, and what do you do to protect your IoT devices from attack?

Your IoT devices are under attack! (Yawn)

Yeah, I get it. For quite some time now I've been hearing about how insecure IoT devices are. And, honestly, I sort of stopped paying attention to IoT malware attacks. What else is new? And besides, that's the kind of thing that happens to other people, right?

Wrong. If you have IoT devices in your home, or on your corporate network (or both), this might have already happened to you. And if it has not happened, it almost certainly will. The truly frightening thing is that your devices might have already been attacked and compromised. And you might not even know.

Now do I have your attention? Good. So how big is this problem? In this 2015 study by HP 100% of the IoT home security devices tested were found to have "significant" vulnerabilities, and all of the tested devices are likely to be a part of any smart home today: smart TVs, home thermostats, webcams, smart locks, and more.

The vulnerabilities included weak passwords, lack of encryption when devices communicate over the network, and account enumeration (that is, using the password reset feature to find valid user account IDs).

Is it any wonder, then, why IoT devices are such frequent targets of hackers and bot-herders, like the ones who launched Distributed Denial of Service (DDoS) attacks against security blogger Brian Krebs on September 20th and U.S. DNS provider Dyn on October 21st of 2016?

If you're like I was before I really dug into this topic, you have questions:

- What does an IoT device look like under the hood?
- What does an IoT malware attack look like?
- What do you do to protect your IoT devices from attack?

In this article, I'll answer these questions.

Anatomy of an IoT device

To understand what makes IoT devices vulnerable to attack, it's worth a detailed look at what's going on under the hood.

You probably have good idea of what the term "IoT device" means, but just so we're on the same page, let me define the term as I'll use it in this article.

An IoT device is a special-purpose device, that connects wirelessly to a network and transmits and receives data over that wireless connection in order to monitor or control a "thing" (which I'll call a Thing from now on).

IoT device hardware

To understand what is at the heart of an IoT device, we need to understand the key characteristics of the underlying hardware that make IoT devices work:

- Data acquisition and control
- Data processing and storage

Essentially, IoT devices contain *sensors*, *actuators*, or both. Sensors acquire data, and actuators control the data or act on the data.

- **Sensors***monitor* Things and provide data about the Thing, whether it be the temperature, light intensity, or battery level.
- **Actuators***control* the Thing through hardware in the device, like the controls in a smart thermostat, the dimmer switch in a smart light bulb, or the gear motors in a robotic vacuum cleaner. The actuators represent the physical interface to the Thing that make it "go," whether it be to turn on the heat, dim the lights, or send the robotic vacuum cleaner to its charging station.

All IoT devices have a way to process sensor data, store that data locally (if necessary), and provide the computing power that makes the device operate.

If data from multiple sensors needs to be coordinated, or if data needs to be stored in flash memory (for whatever reason), it is the data processing component of the IoT device that does it.

IoT device firmware

The firmware that runs an IoT device is the onboard software that sits between the hardware and the outside world, and generally falls into one of two categories: embedded firmware or operating system-based (OS-based) firmware.

Embedded firmware

IoT devices are resource-constrained, so they often use custom-built, embedded firmware, which is another term for the software that runs on the device. In many cases, the only cost-effective solution for device manufacturers is to engage programmers with a deep understanding of the hardware to write embedded firmware to interact with the hardware.

Embedded software engineers have to perform double-duty. In addition to the embedded firmware, they also write the software to interact with the hardware, along with the application software to interface with the device's user, such as the interface to configure the device, for example.

OS-based firmware

As IoT devices have grown "smarter" (read: more complex) — more sensors, greater data processing and storage capabilities, and so on — the demand for more complicated software to manage and exploit the new capabilities has also grown.

Just as the first computers evolved from firmware loaded from ROM to run the computer's basic functions to an operating system like MS-DOS instead, IoT devices are maturing in a similar fashion. An IoT device now probably runs an operating system (OS) that provides an abstraction layer between the hardware and other software that runs on the device.

By providing an abstraction of the underlying hardware from the device's application software, the IoT operating system enables a familiar division of labor. Embedded software engineers (who understand the hardware) can now spend their time writing device drivers, and application programmers (who do not need to understand the hardware intimately) spend their time writing the software that makes the device "smart".

A popular OS choice for many device manufacturers is Busybox, a stripped down version of the Unix operating system that contains many of the most common utilities, has a very small footprint, and provides many capabilities of Unix in a single executable.

IoT device wireless communication

IoT devices most often communicate wirelessly, which means they can be anywhere in your home or enterprise. The communication needs of the device change depending on how it is designed to work.

Some devices are designed to work by making a direct 802.11 Wifi connection to your router. From there, the device can access the internet. A motion-activated security camera is a popular example of this type of device, which uses Wifi because it potentially needs a fair amount of bandwidth.

Some devices are meant to work as part of a group of IoT devices. For example, a window open/closed sensor that is connected to a smart home gateway device (sometimes called a hub) uses a wireless protocol like Z-Wave or Zigbee (or any of a half-dozen others) so it can report that the window has been opened.

IoT device management

You manage your IoT devices in two main ways: you have to connect the device to the network (provisioning), and once it's connected you monitor and control it. I'll explain these more below.

Provision the device

Many IoT devices (especially small ones like a temperature sensor) do not have built-in user interaction hardware, such as a touch screen, and are called "headless" devices. One way to configure headless devices is to use Wifi Protected Setup (WPS), which requires a WPS-enabled device and a WPS-enabled router. In the simplest scenario, you press the WPS button on your IoT device, then press the WPS button on the router, and the two devices are eventually connected.

Some devices will create a Wifi access point to which you can connect by using your smart phone to access a setup program where you to enter your Wifi network credentials.

Other devices, like gateways, scan and add devices that it detects are in setup or pairing mode. You simply configure the gateway so it has internet access, tell it to sniff out other devices, and follow the device-specific instructions to put the devices in pairing mode so that they can connect to the gateway.

Monitor and control the device

Once your device is connected to the network, you can monitor and control it. One way to control it is through a smartphone, either connected to the gateway directly (inside your home, for example) or through an interface to a cloud service.

Some devices like CCTV security cameras connect directly to the internet and have dedicated IP addresses. These devices can be accessed directly over the internet, bypassing the need for a cloud service provider or gateway.

Many IoT devices are installed in homes and businesses, but are exposed directly to the internet by modifying your firewall to enable port-forwarding. This allows the device to be conveniently accessed from anywhere on the internet to monitor and control it.

IoT device security

Lastly, there's security. That's right. Last. Unfortunately, that's the single biggest problem with IoT devices: security is most often last. It's an afterthought.

I get it. It's difficult to create a reliable, resource-constrained device that can connect to a wireless network, use very little power, and is most importantly (to the consumer) cheap. Because there is so much to do to just produce a working device, is it any wonder security is the last thing to be considered in the development lifecycle?

It's worth noting that there are lots of manufacturers out there who take security very seriously, but their devices tend to be pricey. That's the tradeoff.

So, now we have all these cheap, er, I mean inexpensive or *cost-effective*, devices on the network with very little security. This, my friends, is an IoT malware attack waiting to happen.

Anatomy of IoT malware attacks

So we hear about "IoT malware" a lot, but what does that mean, really? Let me break it down, starting with the attacker.

The attacker

Who are these people? The short (and unsatisfying) answer is: nobody knows for sure.

In this article, renowned security expert Bruce Schneier says that based on the scale of recent attacks, the perpetrators are probably not activists, researchers, or even criminals.

According to Schneier, the attacks are designed to test the defenses of the target by employing multiple attack vectors, causing the target of the attack to put up all of its defenses in the process. Schneier says such carefully executed attacks are characteristic of state actors (government body). Now, that's a scary thought, and hopefully Schneier is overreacting a little.

At any rate, it's not crystal clear who the attackers are, but one thing is clear: they're clever, resourceful hackers. Do not underestimate them.

The attack vector

For any type of attack (malware or otherwise), the attacker needs to hit an *attack surface*, which is defined as the sum total of all of the device's vulnerabilities. When the attacker identifies and becomes familiar with the attack surface, they identify an *attack vector*, which is a path the attacker can use to exploit the device, which allows the attacker to make the device do something other than what it was intended to do.

Let's look at some common IoT device attack vectors.

Weak passwords

Aside from security's low priority in the device development lifecycle, manufacturers want their devices to be easy to setup and use. Manufacturers know that many IoT device end-users are not often technically savvy. In order to make the device easy to setup and use, the manufacturer provides some simple way to login to the device, like a single userid/password combination.

This simplicity creates three problems:

1. After the device is set up, the vast majority of users then go about their merry way and leave the device's login credentials unchanged.
2. After the device ships, the default userid/password is added to the list of known exploits for that device.
3. Manufacturers continue to use easy userid/password combinations (for example, admin/admin, user/user, and so forth), or make up new, equally simple ones, which then quickly join the ranks of known vectors.

Lack of encryption

Because security is unfortunately often an afterthought in the IoT device development lifecycle, security features like encryption are often overlooked or not even considered. The industry is requesting embedded cryptography, such as cryptographic co-processors that can handle encryption and authentication in IoT devices. When designing and building your IoT apps, securing your data over the network (a la data encryption techniques) must be part of your design.

Unfortunately, many IoT devices do not support encryption, which means you need to really do your homework when investigating the devices you intend to use as part of your overall solution to make sure they provide encryption.

Backdoors

Some IoT device manufacturers put "hidden" access mechanisms in their devices called *backdoors*. Ostensibly, this makes the devices easier for them to support. But in reality, it might as well open the *front door* for hackers. While most users don't have the technical know-how to crack a backdoor, for a hacker, it's child's play.

No worries though, once a backdoor becomes known, the manufacturer apologizes profusely and immediately releases a firmware update closing the backdoor. Right? You would think so. Unfortunately, there are numerous stories like this one, where a manufacturer has a known backdoor in their device, but rather than remove the backdoor, the manufacturer just made it more difficult to access (or so they think).

In most cases, the backdoor is either a userid/password or an open port on the device (that you can't close). To call these "backdoors" is a mistake. To a hacker, these are wide-open front doors. Plain and simple.

Internet Exposure

Anytime a device is exposed to the internet — meaning that it will accept incoming traffic — it will come under attack. I guarantee it.

Consider this example from my work. I lease a number of virtual servers that I use for running websites, and leave port 22 open so I can SSH into them. With just default firewall rules, these hosts are under constant attack. As in hundreds of login attempts per hour! Of course, I run iptables to set rules on every server I manage to block IP addresses of failed logins for long enough to weaken scripted attacks. So now I see "only" 5-10 failed logins from around the globe per hour.

My point is this: expose anything to the internet, and it will be attacked. And, unlike a hardened server where you can control the firewall and how the host is accessed, most IoT devices have little or no security and are particularly susceptible to attack.

But wait, there's more...

As you can see, IoT devices are rife with vulnerabilities. Weak passwords, coupled with direct device exposure and backdoors, make IoT devices easy pickings for even the least sophisticated hackers (called "script kiddies" by the way).

Think only state actors and the most sophisticated hackers have the skill to hack your IoT devices? Think again.

The Open Web Application Security Project (OWASP) has a sub-project called the IoT Attack Surface Area Project, where they have a list of potential vulnerabilities in the IoT attack surface.

The attack

You've seen how an attacker gets into the IoT device, so now let's talk about the attack itself.

To set the stage, the point of an IoT device attack is to take over the device and bend it to the hacker's will. So the attack comes in two phases: the scan and takeover phase and the attack launch phase. Both phases are normally executed by a Command and Control (CNC) program.

Scan and takeover

The CNC program scans IP addresses on the internet looking for hosts with open ports, and if it finds one, it attempts to log in using a set of known default userid/password combinations (for example, admin/admin, root/admin, user/user, and so forth).

If the login succeeds, a script runs that reports the device's IP address, along with the login credentials to use. The CNC program then pushes the malware to the device that it needs to run the attack. The device is now pwned, and awaits further instructions from CNC to begin the attack.

The attack scanning program continues this process, taking over as many new devices as it can. Each device that has been taken over is referred to as a *bot*.

Nobody knows for sure how much time passes from the moment an IoT device becomes a bot until the time it is used in an attack. It could be hours, days, weeks, or months before a bot is called to action. During that time, the device's owner is almost certainly unaware of what is going on.

Attack launch

A single IoT device is not typically very powerful, and so a single bot is not much of a threat. But create a horde of bots networked together to achieve a common purpose, and, look out! This type of *botnet* attack is made up of hundreds, thousands, and even hundreds of thousands of bots, all under the control of the hacker. Scary.

The attacker typically uses their botnet army for one of two purposes: DDos attacks or spam bots.

- A **Denial of Service (DoS) attack** is designed to cripple a target host by sending it so much HTTP (and other) traffic it cannot handle the volume. Eventually the targeted host fails to respond, effectively taking down the host.
  Rather than originating from a single powerful computer (or cluster of computers), a Distributed Denial of Service (DDoS) attack originates from many host computers (thousands or hundreds of thousands). In the case of an IoT botnet attack, the host computers are the legion of IoT devices. The target doesn't stand a chance.

- **Spam bots** are fueling the spam industry. There is big money in the spam industry. System administrators spend vast amounts of time and energy to blacklist known spam relays, hoping that just a fraction of a spammer's emails make it to your inbox.
  But what if the spam could appear to have been sent from a non-blacklisted IP address? An IoT device running on grandma's network through grandma's router? Perfect! It's not likely that grandma's IP address will pop up on a blacklist. Besides, if grandma's smart TV is just one of a hundred thousand bots sending spam, imagine the administrative nightmare trying to figure out which of all those IP addresses to blacklist!

Once an attacker has a botnet army at their disposal, they have a sea of small devices they can use to create a terrifying flood of internet traffic or spam the world.

Recent examples of IoT malware attacks

Here are some recent IoT malware attacks that you may have heard of.

Mirai

This is a Busybox attack. It works by scanning the internet for hosts with an open port 23 (telnet), and using a weak password vector to gain access to devices that are running Busybox. Once inside, the malware is installed and contacts the CNC server where it awaits further instructions. When attacking, the Mirai CNC server instructs all the bots under its command to launch a flood of various kinds of traffic, overwhelming the target host.

Mirai is possibly the most well-known attack, and (as it turns out) mostly used infected CCTV camera devices to carry it out. Several things make Mirai different:

- Mirai contains a "do not mess with" list of servers that include General Electric, Hewlett Packard, and the U.S. Department of Defense.
- The author of Mirai, known only as "Anna-senpai" on Hack Forums released the source code on September 30, 2016.

Mirai hit in several major waves. The first attack was on security blogger Brian Kreb's site on September 20, 2016. Another attack occurred on October 21, 2016 against US DNS provider Dyn that disrupted the popular streaming service Netflix, along with Twitter, Airbnb, and others.

Security researcher Robert Graham of Errata Security blog presented an analysis of the attack at the 2017 RSA Security Conference in San Francisco, CA, USA.

Brickerbot

Security firm Radware first warned about a potential attack they dubbed "Brickerbot" on April 4, 2017.

Another Busybox-based attack, this malware bricks the device (makes it unusable), hence the name.

In this type of attack, known as a Permanent Denial of Service (PDoS) attack, Brickerbot does this through a series of Busybox commands that wipe everything from the device's internal storage through the Unix `rm` command, along with commands that reconfigure the kernel, and finally reboot the (now useless) device.

Later in April a "gray-hat" hacker whose Hack Forums userid is "Janit0r" claimed to be the malware's author, saying in a HackForums post that the virus was targeted at "careless manufacturers" of devices that are so easily hacked. You can read more about it here.

Spam bots

Email is the lifeblood of spammers, whose real goal is to drive traffic to their customers' websites through emails with catchy subjects, lewd content, and so forth (known as click bait). The tactics employed to bait you into clicking on a link vary ("Lose 100 pounds overnight! CLICK HERE NOW!" or "Get a free iPhone. CLICK HERE NOW!").

The entire strategy hinges on their email arriving in your inbox. The main problem spammers have is sending their emails so they won't be caught in spam filters, many of which use "blacklists" of Simple Mail Transport Protocol (SMTP) server IP addresses known to be used by spammers (like open relays).

However, if a spammer could use a legitimate-looking *proxy* to their SMTP server — called a SOCKS proxy — whose IP address isn't blacklisted (remember grandma's smart TV?), their spam emails have a greater chance of finding their target (but you're still not getting a free iPhone, sorry).

The Linux.ProxyM virus is a *secondary payload* Trojan, which goes to work once the initial Trojan has infected your computer. How? Something like this: you're baited into clicking on that "get your free iPhone now!" link, and you agree to install the "Simple free iPhone plugin" (the initial Trojan), which infects your computer. At that point a script goes to work, which scans for vulnerable IoT devices. If it finds one, it uses the now familiar weak credential exploit to gain access.

Once access to the device is gained, it is infected with the secondary payload containing the actual malware that drives the attack. This malware connects to a CNC server that provides a list of email addresses, and an SMTP server. At that point, now acting as a SOCKS proxy, your device sends spam emails at the behest of the CNC server.

How to protect your IoT devices

So, how do you protect your IoT devices from being infected? You don't allow them to become infected to begin with. I know, really helpful advice.

If you already have devices deployed, I have good news and bad news. The bad news is that if your devices are directly exposed to the internet (as I described earlier), they have at best been probed, and at worst, have been turned into bots.

The good news is that most IoT malware resides in memory, so as long as the device is powered on, the malware is alive. Reboot the device and the malware is gone.

All kidding aside, it's still best to prevent your devices from becoming infected to begin with. Here are a few tips, courtesy of Captain Obvious.

Always change default passwords

When you provision a new device, always change the default password. This seems so simple, yet in the hustle and bustle of setting up a new device so you can play around with it — er, I mean, put it to useful work — it's easy to skip this vital step. *Don't skip this step!*

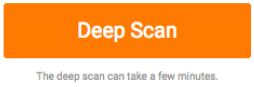
Go into the management interface and change the password. And, if there is not a way to do this, and you plan to expose the device to the internet, send the device back. You can be optimistic, of course, and hope the device doesn't get turned into a bot, but malware writers love optimists.

Remove devices with telnet backdoors

Vendors may think themselves clever by putting these backdoors in, but they're not. It may make support easier for the manufacturer, but at what cost to you?

A device with an open telnet backdoor should be removed from the network, but how do you know? There are IoT device scanners like this one from BullGuard, which scan an IoT search engine called Shodan to reveal if your devices are vulnerable based on the IP address of the computer where you originate the scan. *Please note: as a rule of thumb, only scan IP addresses that are yours or that you have the owner's permission to scan!*

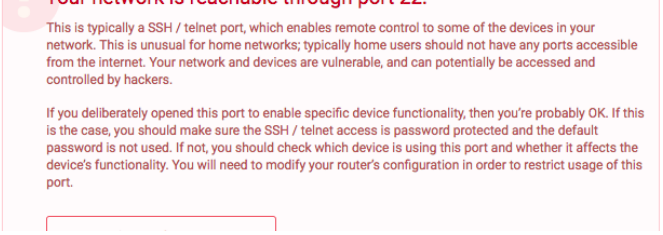
Here is a scan I ran from my computer.



The deep scan can take a few minutes.

®

If the scan looks like this, you may have a problem:


Your network is reachable through port 22.
This is typically a SSH / telnet port, which enables remote control to some of the devices in your network. This is unusual for home networks; typically home users should not have any ports accessible from the internet. Your network and devices are vulnerable, and can potentially be accessed and controlled by hackers.
If you deliberately opened this port to enable specific device functionality, then you're probably OK. If this is the case, you should make sure the SSH / telnet access is password protected and the default password is not used. If not, you should check which device is using this port and whether it affects the device's functionality. You will need to modify your router's configuration in order to restrict usage of this port.

Never expose a device directly to the internet

When you are faced with the question of whether or not to expose a device to the internet by opening up your firewall, the right answer is almost always **no**.

BullGuard provides a way to do a "deep scan" to check for any open ports on your publicly exposed IP address assigned by your ISP. This allowed me to see if I had any open ports on my router. I was relieved to see that I did not.

Tell them about the free scan.

Facebook Twitter Google+

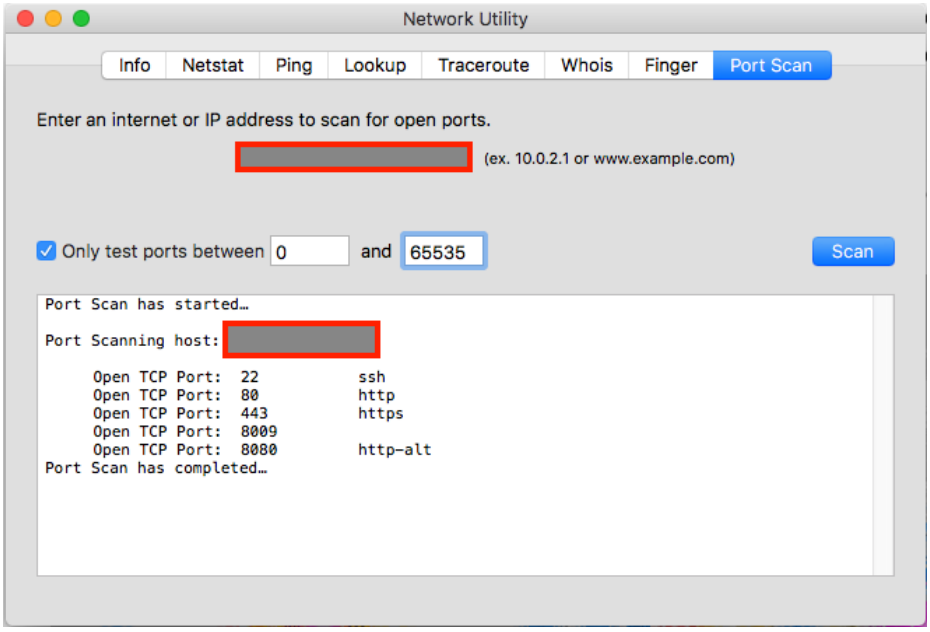
®

Run port scans on all your machines

Okay, so scanners like BullGuard can give you a level of comfort that your IP address is locked down, but if you're like me, you want to run the tools yourself.

I used the Mac Network Utility to do a scan of one of the virtual hosts I lease to see which ports were open.

Figure 4. Mac Network Utility scan



These results were not really a surprise to me. This virtual server hosts a website, running Apache, with a Tomcat AJP backend, and SSH access for admin purposes.

Conclusion

In this article I showed you a detailed look at the anatomy of an IoT device. Then, I showed you the anatomy of an IoT malware attack and some malware attacks that have made the news. Finally, I showed you how to protect your IoT devices by changing default passwords and running scans of your IP address.

Related topics

- Securing IoT Devices and Gateways
- IoT Security Best Practices
- IBM POV - Internet of Things Security