April 2015
Geoff Huston

The Internet of Stupid Things

In those circles where Internet prognostications abound and policy makers flock to hear grand visions of the future, we often hear about the boundless future represented by *The Internet of Things*. This phrase encompasses some decades of the computing industry's transition from computers as esoteric pieces of engineering affordable only by nations, to mainframes, desktops, laptops, handhelds, and now wrist computers. Where next? In the vision of the *Internet of Things* we are going to expand the Internet beyond people and press on with connecting up our world using billions of these chattering devices in every aspect of our world.

It's not a new vision by any means. Already my car probably has 100 microprocessors doing everything from regulating the engine to remembering the seat position. But this grand vision connects all these processors up in one massive Internet. Gartner have projected that the world of chattering silicon will get to 25 billion devices by 2020. Cisco has upped the ante with their prediction of 50 billion such connected things by 2020, and Morgan Stanley has trumped them both by going further with a prediction of 75 billion devices connected to the Internet in that time. Other reports have placed this number as high as 100 billion. The extent of the current levels of unbounded technical euphoria in this space project economic values of this activity in units of trillions of dollars by 2020.

What do we know about the "things" that are already connected to the Internet?

Some of them are not very good. In fact some of them are just plain stupid. And this stupidity is toxic, in that their sometimes inadequate models of operation and security affects others in potentially malicious ways.

One of the earlier incidents of stupidity on a grand scale occurred in 2003 when Dave Plonka, then at the University of Wisconsin-Madison, reported on what appeared to be a large attack directed to a time server located on the campus. As was reported at the time:

> "In May 2003, the University of Wisconsin - Madison found that it was the recipient of a continuous large scale flood of inbound Internet traffic destined for one of the campus' public Network Time Protocol (NTP) servers. The flood traffic rate was hundreds-of-thousands of packets-per-second, and hundreds of megabits-per-second.
>
> Subsequently, we have determined the sources of this flooding to be literally hundreds of thousands of real Internet hosts throughout the world. However, rather than having originated as a malicious distributed denial-of-service (DDoS) attack, the root cause is actually a serious flaw in the design of hundreds of thousands of one vendor's low-cost Internet products targeted for residential use. The unexpected behavior of these products presents a significant operational problem for UW-Madison for years to come."
>
> http://pages.cs.wisc.edu/~plonka/netgear-sntp/

The equipment, a Netgear CPE modem, used the SNTP protocol to pick up the time of day from the network, as it did not have an internal battery or clock, and it hard-coded the IP address of the network time server operated by the University of Wisconsin into the product. The more units that were sold the greater the aggregate traffic volume that was sent to the university's time server.

Was this a fixable problem? Could the vendor perform a product recall?

> "Both Netgear and other members of the review team felt that it was unlikely that all but a very small subset of the owners would return the affected device since they appear to be working fine. Also, very few customers have registered these products with the manufacturer, so it is impractical to contact them."
>
> http://pages.cs.wisc.edu/~plonka/netgear-sntp/

It's a "thing" that sits in a cupboard somewhere. It only gets looked at when it stops working. There was no way to upgrade the device's software remotely, and no way to alert the human user that they needed to upgrade the device. So the problem persisted.

In this case there was one victim of this stupidity. Other forms of toxic stupidity affect many more people. The website http://openresolverproject.org describes the ongoing efforts by Jared Mauch to document the extent to which devices have been deployed on the Internet that are in effect "open" DNS resolvers.

These devices are generally CPE modems that interconnect a local network to an Internet service. They were probably intended to operate in a more restricted sense, answering DNS queries coming from the "inside" by sending queries to the "outside". But it appears that in some cases the software failed to make the distinction between "inside" and "outside", and answered queries coming from the "outside" as well. There are many of these stupid devices out there. Some 28 million of them in October 2013 according to this web site's census.

These devices can be co-opted into behaving in a toxic manner by sending them DNS queries over the Internet. The open resolver will attempt to answer the query (because it doesn't know the difference between "inside" and "outside") and send the response to the source address in the query. The DNS response can be contrived to be significantly larger than the query. One form of toxic attack is to coopt all of these resolvers to ask a query from a single DNS authoritative name server, in an effort to swamp the name server. Another form of toxic attack is to generate queries with a fake source address, namely the IP address of the intended victim and cause many millions of these open resolvers to all direct this DNS response traffic toward the victim.

As with the earlier problem with SNTP and Netgear, there is no easy fix here. These are unmanaged devices that just sit in a cupboard. Nobody is looking after them.

Here at APNIC we have been affected by a similar problem. One of our servers was experiencing a continuous load of some 5,000 queries per second, all for a single script that returned the IP address where the query came from. It was just another simple "what's my IP address?" script. When we wanted to migrate servers the question of exactly why this query load had appeared came up. It seems that like the earlier experiences with poor quality control in CPE controller software, a manufacturer had embedded the URL for the APNIC "What's my IP address" script into its software for some form of DVR or television. And for some reason the software in the device used this query regularly as part of its operating procedures.

It seems that only APNIC was the victim here. However, maybe it's a little deeper than that. Certainly APNIC could now tell from these queries how many of these units were being sold, and where they were sold. However its more than a leak of market intelligence, as it raises more uncomfortable questions as well. What if the APNIC script was altered gave back the wrong IP address? What if the script froze the TCP connection and never answered? Would the device crash if the script gave back a few megabytes in response rather than a few bytes as expected. Could the response overrun the input buffer and rewrite the operating stack inside the device? Could the script's response be used to open up the device and turn it into a remotely controlled bot? Obviously we're not going to do any of that that. But it's not a comfortable position knowing that we are now on someone else's critical path.

Maybe it all gets better when the software is used in a device that is "managed". Last year 12% of all the smartphones sold on the world, or some 180 million units, ran Apple's IOS software. Part of Apple's business model includes the App Store, and to guide users through this store the iPhone is "locked". But any search engine will happily direct you to directions on how to unlock your iPhone, exploiting bugs in the IOS software across most version of IOS. So its not just the software used in low value unattended equipment that has its problems, but even on high volume high value equipment where there is a strong commercial motivation to deploy the best quality software.

When we think of an *Internet of Things* we think of a world of weather stations, web cams, "smart" cars, personal fitness monitors and similar. But what we tend to forget is that all of these devices are built upon layers of other people's software, and assembled into a product at the cheapest possible price point. It may be disconcerting to realise that the web camera you just installed has a security model that can be summarised in a single word: *yes*, and its actually offering a view of your house to the entire Internet. It may be slightly more disconcerting to realise that your electronic wallet is on a device that is using a massive compilation of open source software of largely unknown origin, with a security model that is not completely understood, but appears to be susceptible to be coerced into being a *yes*.

What are we going to do about it?

It would be nice to think that we've stopped making mistakes in code, and from now on our software in our *things* will be perfect. But that's hopelessly idealistic. It's just not going to happen. Software will not be perfect. It will continue to have vulnerabilities.

It would be nice to think that this *Internet of Things* is shaping up as a market where quality matters, and consumers will select a more expensive product even though its functional behaviour is identical to a cheaper product that has not been robustly tested for basic security flaws. But that too is hopelessly idealistic.

The *Internet of Things* will continue to be a market place where the compromises between price and quality will continue. The concern is that we may sacrifice quality in order to support a low price. If that's the case then what's going to stop us from further polluting our environment with a huge and diverse collection of programmed unmanaged devices with inbuilt vulnerabilities?

What can we use to make this world of these cheap things less stupid and less toxic?

Comprehensive answers for this question are difficult to come by.

However, I think we already know some parts about what it will take to avoid an Internet full of toxically stupid things.

One part of any useful answer is that we really should use IPv6 for the *Internet of Things*.

There are many problems with IPv4, and one of them is that the address space is just too small. If you want your thing to be visible on the Internet, then everyone else can find your thing as well.

> "ZMap is an open-source network scanner that enables researchers to easily perform Internet-wide network studies. With a single machine and a well provisioned network uplink, ZMap is capable of performing a complete scan of the IPv4 address space in under 5 minutes, approaching the theoretical limit of ten gigabit Ethernet."
> https://zmap.io

Yes, just 5 minutes to scan the entire IPv4 space. On IPv4 there is nowhere to hide. If you want your *thing* to be visible on the Internet then everybody else can see it too.

What about IPv6? If it takes 5 minutes to scan 4 billion addresses, how long would it take to scan all of IPv6's 340 undecillion addresses? Well at that speed it would take 70 sextillion years! Even with the usual forms of pruning and heuristics to guide the search, scanning the IPv6 space is not a realistic proposition. Even the 64 bit interface identifier field takes 41 thousand years to scan at this speed, so a IPv6 system that used appropriately random privacy addresses would still be hiding behind a massive search space.

Another part of any useful answer is to avoid unnecessary external dependencies. For example, try not to betray your existence by invoking external scripts. Don't rely on some other set of resources being online and available.

And of course its always useful in this world to be paranoid. Don't trust the Internet. That's not the same as don't use the Internet. By all means leverage its connectivity and its services, but don't be overly credulous. If you are using the DNS to map resource names to IP addresses then use an internal DNS library that validates what it hears, rather than relying on some external resolver not to lie to you. If you use secure channels to access the device, and you probably should, then use a security model that is tightly focussed on the use of a specific trust anchor to seed trust. Think of the Internet as a hostile space: How can you stop your thing from being captured and exploited by others?

Nothing is perfect. Software changes faster than hardware. Think hard about how you want to maintain these devices once they are deployed in the fields of the Internet. How can you upgrade their software to correct a calamitous bug?

None of these are new lessons. Were we able to retrospectively apply these lessons then we would not be staring at some 30 million unmanaged open DNS resolvers wondering just what we are going to do about them. But that's already done. More importantly, what we need to avoid is further expanding the pool of these toxically stupid things. If we are heading towards an Internet of hundred billion things over the next five years, then let's make sure that we remember past mistakes and learn from them. We really need to ensure that its not an Internet of 100 billion stupid things that can be readily co-opted for evil.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of a number of Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005, and served on the Board of Trustees of the Internet Society from 1992 until 2001.

www.potaroo.net