KEYNOTE ADDRESS: THE BATTLE FOR A SAFER INTERNET*

VINT CERF**

SILICON FLATIRONS CENTER
BOULDER, COLORADO
FEBRUARY 11, 2018

As agent provocateur, I would like to draw to your attention several issues about the unsafe Internet that we are living with today. I accept a certain amount of responsibility for some of that lack of safety. Some of it, in many cases, is almost unavoidable. But that does not mean that we should not do something about it. The fact that we are here, in the law school, is important because some of the things that we need to do involve the creation of legal frameworks and law enforcement and the like, in order to create a safer environment for all of us. If we do not feel safe using the Internet then we are not going to want to use it. If we don't use it, then we will lose a lot of the benefits that we have already encountered in this environment.

I am deeply interested in this mantra of safety, privacy, and security, in cyberspace. I want to emphasize how shared the responsibility is for creating safety, privacy, and security. This is not just the responsibility of the private sector and the products and services that it offers. And it is not just the responsibility of law enforcement agencies, and the like. It is you and me, behaving in ways that are responsible. We cannot escape some personal responsibility for behaviors that will help make the Internet a safer place.

---

* This speech has been edited for publication. Vint Cerf delivered these remarks at the Silicon Flatirons Regulating Computing and Code Conference at the University of Colorado Law School on February 11, 2018. The focus of Cerf's keynote address was the safety, security, and privacy of the Internet as we look further into the 21st Century. For a video of the speech, see Silicon Flatirons, *Sunday: Introduction and Keynote*, YOUTUBE (Feb. 12, 2018), https://youtu.be/atWFGNlmiKg [https://perma.cc/E4JZ-A6PN].

** Vint Cerf, now vice president and Chief Internet Evangelist for Google, is widely known as one of the "Fathers of the Internet." Cerf is the co-designer of the TCP/IP protocols and the architecture of the Internet. He contributes to global policy development and the continued spread of the Internet. For a more complete biography, see *Vinton G. Cerf*, GOOGLE AI, https://ai.google/research/people/author32412 [https://perma.cc/6XEQ-TAJA].

What I think is particularly important is that there needs to be a certain amount of liability for bad behavior, harmful behavior, or irresponsible behavior. Consider how we have induced better personal behavior in the past—seatbelts, are a good example. For many years we kept telling people that seatbelts would keep them from ending up looking like "this," and we would show people pictures of horrible crashes. But that wasn't really enough. We also had to say, "By the way, if we catch you driving around without a seatbelt, there will be consequences."

Smoking? Same kind of thing. We told people all about how bad smoking was, but in effect, we also had to say, "By the way, if you smoke in this building, there will be consequences." So the law and law enforcement have potential roles to play in achieving commonality in safety, privacy, and security. In my opinion, as a former software developer, the heart of many of the problems that create a lack of safety, or a lack of privacy, or a lack of security, is buggy software. It's embarrassing to say or to admit that over the past eighty or so years—when people have been writing programs for computers—that we have never figured out how to write programs that have no bugs.

There are bits and pieces of light, here and there, but for the most part almost every piece of software that I have ever encountered has a bug in it somewhere: a bad assumption, or a failure to look at all the possible corner cases—there are all kinds of reasons why there are bugs. What this means is that we have to assume that software will have bugs, so then we have to ask: "How are we going to cope with the resulting problems?"

Some of those problems may simply be annoying, I know. On the other hand, a failure of some pieces of software could be literally fatal: a self-driving car that doesn't know how to cope with a particular situation runs into the nearest tree and kills the occupants, for example. I also need to point out that the network is not the origin of these problems. Even before we had Internet or other networks, people were running around, trading diskettes back and forth. Those diskettes often had pirated software on them, but they sometimes contained malware too. People were spreading malware around to various computers, independent of the Internet, just by trading media diskettes.

Today it's, of course, flash drives and things like that, but there is also the quality of the programming environment in which we create software. There, tools are not necessarily very helpful to call to the programmer's attention the mistakes that have been made in logic. For example, I have always wanted to imagine that there is this little software thing that sits on my shoulder, watching while I write code, saying, "You just made a buffer overflow mistake." And

I say, "What do you mean?" And it says, "Well, look at line number 123."

I would actually like to have a software environment—even if it is not quite that personable—where I can at least say, "Can you find any references to variables in the program that I didn't set first?" Otherwise, I am getting a random value and making a decision based on that which is most of the time the wrong choice. "Can you help me uncover stupid mistakes like that?" If anybody here has ever made a living writing software and debugging what they have written, a lot of it is forehead-slapping: "Boy, that was stupid. How could I have done that?"

Part of the story here is that we need better tools in order to avoid some of the problems. This tool argument also goes for people who are trying to protect themselves in the online environment. We need to provide users with the tools to do this (and we will come to some of that in a bit). But there is one other thing that you might not think of unless you have been in the software and hardware business. You might not know that, many years ago, in the 1960s, MIT undertook a project called Project MAC—had nothing to do hamburgers—it had to do with Multi-Access Computing.[1]

This was a large scale time-sharing system.[2] Part of the design of that system included some hardware modifications of the computer that they were using.[3] One of these modifications involved what we call "rings of protection."[4] What this meant was that while the computer was executing software, certain instructions were not permitted.[5] The computer would block the execution of certain instructions that potentially would let you store information in places where it should not go, like in the middle of the operating system.[6]

There were eight levels of privilege.[7] If you executed an instruction and you were in the wrong level of privilege, you would be trapped down to a little piece of software called the kernel, and it would say, "Who the hell are you and why do you think you can execute this instruction?" Unless you could show the proper bona fides, it would inhibit and prevent you from executing the instruction.[8] If you could move into a new level of protection, one

1. Tom Van Vleck, *Project MAC*, MULTICS, https://multicians.org/project-mac.html [https://perma.cc/DM57-K28F].

2. *Id.*

3. *See id.*

4. *See* Tom Van Vleck et al., *Glossary of Multics Acronyms and Terms: Ring*, MULTICS, https://www.multicians.org/mgr.html#ring [https://perma.cc/A78A-TYYP].

5. *Id.*

6. *Id.*

7. *Id.*

8. *See* Tom Van Vleck et al., *Glossary of Multics Acronyms and Terms: Ring Brackets*, MULTICS, https://www.multicians.org/mgr.html#ringbrackets [https://perma.cc/RN86-U7DA].

that gave you that privilege, then you could execute the instruction.[9]

The idea behind Project MAC was to prevent random software from causing damage to the system by using the hardware to reinforce the software protections that the system provided. Ironically, all of the Intel x86 chipsets actually have those features in them, but nobody has been programming to use them. I submit that we should be paying attention to using the hardware to reinforce the software protections that we're putting into these systems.

Another example is when a computer is booting up. It's typical that a computer loads the operating system in right at the very beginning. This is the most vulnerable moment in the life of a computer. One thing that you do not want to happen is that the computer starts out by booting in a piece of malware, a version of the operating system that's harmful. One way to try to defend against this is to use digital signatures, which I am assuming most of you have at least heard of. This basically says, "This software is recognizable as having been provided by a known acceptable source." The hardware checks the digital signature on the operating system before it boots it in and refuses to start if all it can find something that does not meet that requirement. Again, it is hardware reinforcing things.

There are a lot of folks who have been working on what is called a trusted computing base, which is a core piece of hardware and software that has a footprint of software small enough so that you can actually say something, with confidence, about its correct operation. The trusted computing base is used, for example, as the place to hold keys for cryptographic operations or for checking digital signatures. The trusted computing base uses both hardware, that you believe is protected from tampering—that is not easily penetrated—and software methods in order to create a safe place in a more general operating environment.

Another thing that you and I can take advantage of is called two-factor authentication. We use this at Google—it is required that Google employees have two-factor authentication dongles that we carry along with our badges. They plug into the USB ports, or sometimes it is a program that is running in a mobile format that either uses Bluetooth, or near-field communication (NFC), or something like that, to provide an additional key beyond just a username and password.

The theory here is that if somebody finds out what your username and password are, even by just guessing or by "dictionary attacks," they still cannot get in because they do not have the

9. *Id.*

second factor. This device is openly available—we offer it to all of our users. As far as I know, however, maybe only ten percent of them use it—because it's inconvenient. You have just entered your username and password and now this thing wants something else. The trouble is, people pick really bad passwords. Some people pick "password" as their password because it is easy to remember, except the bad guys know that too.

Even if you pick a really good password using computer power these days, even if you one-way encrypt the passwords, people can use what is called a "dictionary attack" to discover your password. These people encrypt every word in the dictionary and then compare the encrypted dictionary term(s) with the encrypted passwords that they managed to pull out of somebody's machine. As soon as they find an encrypted match, they know what the plaintext word was in the dictionary, they know your password. End of story.

So two-factor authentication is great. However, I have about 300 passwords for accounts scattered around the Internet, so even if I wanted to have two-factor authentication, if I had to have a dongle for each password, I'd have 300 of these things—I'd need a big bag to carry the darn things!

There has to be some architecture in these devices that allows you to put lots and lots of different cryptographic passwords into the same device. That requires standardization. We are far away from agreement on the specifics. But again, two-factor authentication is a good example—it's a great idea, but you have to think about the scale that people need in order to make it convenient and useful.

There is another thing that we do all the time at Google: monitoring of all layers in the architecture of our distributed systems. We monitor all the way down to, "Is there a power applied to the devices?" And all the way up to, "What is the behavior of the applications?" We know in a system of very large scale that something is always broken, so it is not like we are trying to avoid any breakdowns at all. We just have to know what the normal rate is at which stuff breaks. We're constantly monitoring what's going on. We know when we send error messages to people—the software knows that you just encountered an error.

We monitor the rate at which these errors occur. And when the rate gets bigger than we think it should be, the alarm bells go off, and the teams scramble to figure out why the system is showing this level of failure. Often it is because we pushed a piece of code out that had a bug in it, so then we have to go through the process of figuring out what the bug is, pulling the software back, and pushing out a correction for it. We need a lot more of this kind of design and operational practice in order to improve the safety and

security of the net. Cloud computing is something which is, in a way, a reprise of the earlier days of time-sharing. We used to have these great big time-shared machines that tens of thousands of users could use. Then we went into this period where we had smaller and smaller departmental machines until finally, we had desktops, and laptops, and tablets, and mobiles. Now we've taken all that technology, thrown it into a giant building, and called it a warehouse. That is essentially cloud computing, which is simply large numbers of warehouses, scattered around, all connected by the Internet. It's a very powerful environment because if you're going to fix a bug in the software of a cloud-based system, you can fix it throughout the entire cloud.

This is different from having to send updates to every individual whose laptop has a bug in it. Getting everything to change at the same time is tremendously better; it's more uniform. You can fix things faster, of course. You can also infect everybody faster because if you send out bad code everybody gets hit. But the whole point here is that we can do a better job of protecting the systems because of this uniform environment.

Without going into all the details, I will tell you one other thing that's really tough. One of the things that Google and others are trying to do is to make sure that we don't have only one copy of the important objects that you've entrusted to our care, whether it's your email, or documents and spreadsheets. So, we replicate this stuff across each data center and across data centers, which has required us to build a BFN[10] to connect all the various data centers together. We keep copying data back and forth.

It's hard to make sure everything stays in sync so that we don't have different versions of things scattered around. The thing that I have to give credit to systems engineers at Google for is that, when we're editing documents, for example, it's possible to allow multiple parties to have access to the same document and they can edit it at the same time. There are some little inhibitions. For example, two people can't change the same spreadsheet cell at the same time, but otherwise multiple parties can modify documents, spreadsheets, and presentations concurrently.

This dynamic ability to modify things in a collaborative group is astonishing to me: not only that you can allow multiple changes but also that, when they get distributed, the documents are protected from loss, they're consistent. That is hard. But somehow they have been able to figure out how to do that. There are lots of reasons why cloud computing has become so attractive. It has a cost profile that is pretty impressive. It doesn't take very many people to run a very large-scale data center, partly because of all the

10. Big F****** Network.

monitoring, remote control, and automated systems that are part of it.

At Google, in particular, we do some very serious backup exercises. A lot of people will do desk-type exercises for disaster; we don't do desk exercises. We actually shut down the primary systems and run on backup—live. We do this for a week, and we do this every year. It's nail-biting time. But the fact is, we know whether or not our backup systems actually work. I apologize if this sounds like a sales pitch for Google. I'm intending it to show you the kinds of things that technologists can do to try to make systems more reliable and safer for people to use.

So what about safety? Most of us have computers. We use them in various forms. We have laptops, desktops, mobiles, and so on. But unless you happen to be a big company with a large complement of software people, it's hard to know where to turn for help when things go wrong. If you're a small business, for example, and you run into a problem because your systems are being attacked from the outside for whatever reason, there's often no place to go.

Maybe you bought software that is trying to detect malware in your systems, but which doesn't always work. The day zero attack[11] shows up, but the malware detection doesn't catch the attack because it didn't know about that particular bug. So, I keep thinking, I wonder if we should have a cyber-fire department so that a small business can call for help if their business is on cyber fire, so to speak. It's a very appealing metaphor, but you have to think your way through the scenarios to make sure that it's not abused.

For example, I have Jason over here. He runs a company. Then there's Lindsay. She runs a competing company. So, Jason, being a clever technologist, as he is, calls the cyber-fire department and says Lindsay's company is on cyber fire. The cyber-fire department comes running and disrupts all of her operations, trying to figure out what's wrong. Meanwhile, Jason's operations are fine and he's taking all of her business away from her.

We know that, in a real fire, it's okay for your neighbor to call the fire department and have them come out, smash the windows, and pour water into everything to put the fire out because the fire is a neighborhood hazard. We accept this damage, and we respect this privilege of the fire department to do that. It's not clear that the analogy should hold here. So we have to be super careful, as you all must know, about using analogies in order to reason.

But there is still something there that we should think about. If we don't have a place for people to turn to, then they are going to

11. I.e., a computer attack never before seen in the wild.

suffer various and sundry kinds of failures or harms that we should be trying to protect them from. There is another idea I've heard from time to time that there should be an Internet driver's license.[12] I don't really think that we should force people to take a test and then register them as being allowed to use the Internet. I think that's probably a ludicrous, un-implementable idea.

However, think about what we might do. We take kids in high school and we require them to take driver's training classes, whether they're going to get a driver's license or not. Oh, by the way, unlike my generation, which was back in the previous century, a lot of the millennials and the kids who are in their teens aren't interested in getting a driver's license because they're assuming by the time they are old enough, they can just get into self-driving car, which is sort of, "Wow, very gullible," even though we're hoping that that will be true! Alphabet, the owner of Google, has a company called Waymo that tries to do that.

But still, think about an Internet driver's license, which would teach you about potential hazards—the problems and harms that could occur if you're not thoughtful about the way in which you use the Internet. This could include teaching people to think critically about what they see and hear. I think that children, as they grow up and are educated, should actually learn and be taught how to think critically about what they're seeing and hearing. They should ask questions like, "Where did this stuff come from?" "Did the party who put it into the system have an ulterior motive?" "Are they trying to convince me of something for which there is no corroborating evidence?" This is the kind of scientific-like thinking that I believe everyone should have some training in because we should treat all information sources with a certain degree of suspicion.

But that takes work, and not everyone is willing to put that level of work into analysis and evaluation. Moreover, sometimes there are theories which are later proved wrong, or there are ideas which are wrong but are very appealing to some people when they don't want to abandon some belief that certain things were true even if they're not. That sort of mitigates against trying to get people to use critical thinking. I've even encountered a situation where some families have rejected the idea that children should be taught critical thinking because they come home and are critical of their parent's views. Some parents do not like that.

I still think, however, that as a society, if we want to cope with some of the ills that we discover in this Internet environment—this

12. *See* Tim Cushing, *US Government Begins Rollout of Its 'Driver's License for the Internet,'* TECHDIRT (May 5, 2014, 9:57 AM), https://www.techdirt.com/articles/20140503/042644271106/us-government-begins-rollout-its-drivers-license-internet.shtml [https://perma.cc/5HN2-WT62].

rich source of online content—that critical thinking is a very, very important skill that everybody should have. By the way, that will take care of the other sources of information that we have: radio, television, magazines, newspapers, your friends, and so on.

There are people who say, "Well, can't we use algorithms to sort out the good stuff from the bad stuff?" I wish that were true. In some cases, you can use computer-based algorithms to say "This stuff is sort of outside of the reasonable truth." But the problem is that algorithmic mechanisms rely on the rest of us using the system in order for the algorithm to determine if there's consensus on whether something is true or not. The trouble with this is that the computers that we use to build Internet applications don't know the difference between the computers interacting with those programs and the human beings behind the computers. People create botnets that pretend to be human, and the software cannot tell the difference. When everybody says "thumbs up" on a particularly bad idea, it might be a botnet that is essentially voting up and causing an algorithm to misunderstand that everybody thinks this is consensus when it really isn't. Algorithms are not going to solve all the problems because systems can't distinguish humans from other computer programs.

What about security? I've come to the belief, not a conclusion, that there is an irreducible level of inconvenience associated with good security. It is just inescapable. I do not know what that irreducible level of inconvenience is, but I can tell you that every secure system I've ever dealt with has been inconvenient in one way or another. We have to accept that inconvenience if we want to be more secure than we are today. So, how do we make it easier for people to adopt good security practices?

Well, one thing I thought of is a cyber-hotline between countries that are hacking each other. The US, and China, and Russia are terrific examples, or maybe even North Korea. I think that it would probably be useful to have such a cyber-hotline for this simple scenario: if you're in authority and believe the country is under attack, or the country's interests are under cyber-attack, before you decide to launch a response, either a cyber-response or maybe even a kinetic response, I think it would be good to say, "We think this is happening, we believe you're the source, and we should talk about this before we actually do something."

Anyone who's reading headlines in the last few months knows that this is very important. There's a big issue there. We had the Hawaiian incident,[13] which scared a lot of people into thinking

13. Referring to the January 13, 2018 incident where an emergency alert of an incoming ballistic missile attack was mistakenly sent to cellphones across Hawaii. Panic ensued until the alert was shown to be false and revoked 38 minutes later. *See* Adam Nagourney et al., *Hawaii Panics After Alert About Incoming Missile Is Sent in Error*,

there was something bad actually about to happen. So, I think this will be an important kind of dialogue on which to have a basis to discuss something.

There is also an attribution problem. Cyber-attacks don't necessarily come from where they look like they're coming from. It's possible to have a false flag attack. And the bad part about that is: if you choose to respond against the wrong party, there could be very serious consequences ranging from an international conflict to destroying the desktops or laptops of a whole bunch of people whose machines had been become part of a botnet. They didn't know this. They were not a party to any of the damage done, but their machines just got wiped out.

Not only would that be personally harmful, but those people might actually be people we rely on to do things for the rest of us in our society. They can't do it because their machines got wiped out. So, we have to be very careful about how we respond and how we automate any kind of response to hacking attacks.

Now, ironically, in the original Internet design, I assumed that every machine would have to defend itself because every machine was supposed to be able to communicate with every other machine. We didn't have firewalls in the architecture for very good reason. We have to have machines with software that is fairly suspicious. I even tried to get Google to rename "Android" "Paranoid." I didn't succeed; the marketing people said it was a bad idea. So, we have these various kinds of attack, and we have to be very thoughtful about how we respond.

Let me just summarize on the Internet of Things. There are all those ills that I was describing before that are implicated in the Internet of Things because there are billions of devices with software in them, some of which have bugs. We are handing authority over to the software to do stuff for us on our behalf without our intervention, and the mistakes in the software could cause all kinds of bad things to happen. On top of that, you may have a house full of Internet of Things. These are devices that are programmable that can use the Internet to communicate with each other.

If your house is disconnected from the Internet, you don't want your house to stop working. So, there are all kinds of issues like that, that say that design for this Internet of Things is very important, not the least of which is making sure that the system can distinguish the parents and the owners of the house from the guests. It gets really complicated if you have a fully automated house. How do you introduce the guests to the house? How does the guest know how to turn the lights off and on? How do you

N.Y. TIMES (Jan. 13, 2018), https://www.nytimes.com/2018/01/13/us/hawaii-missile.html [https://perma.cc/9CND-KUJQ].

"unintroduce" the house to the guests when the guests go away, so they aren't doing things to the house when they're not there? You can see the complexities here.

So, I'm going to skip through the Internet of Everything except to say that strong authentication mechanisms—like digital signatures, certificates, and the like—are at the heart of trying to protect ourselves from the invasion of these programmable devices by outside people who should not have access to control our devices.

With regard to security practices, I very much worry that we get things, for example, from the National Institute of Standards and Technology, which is responsible for describing security practices for government agencies.[14] The advice that you get is pretty high-level, and what's missing is the ability to measure how well it was implemented. This creates a measurement issue: knowing whether we've done a good job of following the recommendations that come from there.

I must say that every time I screw up, it's almost always because I made an assumption that was wrong. It's very, very hard to discipline yourself not to make assumptions, and yet I'm finding that I have to make myself remember not to make any assumptions, otherwise I'll make an assumption that turns out to lead to bugs in the software.

We really have to get our security act together. The private sector needs to provide users with better tools, and it needs better incentives to do so. We have to figure out how we get the private sector to invest more in privacy, security, and safety practices. There's a lot of talk about cyber-insurance. Don't be misled into thinking that buying a cyber-insurance policy provides any better security. It only has to do with your financial liability; it doesn't do anything about bad bugs, attacks, and everything else.

And finally, I think that there should be liability and consequences for bad practices. Up until now, most software developers who've gotten away with murder do so by saying, "Well, it's just a bug." At some point, a bug isn't just a bug. It's a serious problem and there should be consequences for having introduced bugs. It could've been avoided.

14. *See NIST Mission, Vision, Core Competencies, and Core Values*, NAT'L INST. STANDARDS & TECH., https://www.nist.gov/about-nist/our-organization/mission-vision-values [https://perma.cc/U63S-Z3K8].