



[www.computer.org/internet computing](http://www.computer.org/internet-computing)

Aspects of Internet Security

Barry Leiba

Vol. 16, No. 4
July/August, 2012

This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

IEEE  computer society

© 2012 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

For more information, please see www.ieee.org/web/publications/rights/index.html.



Aspects of Internet Security

Barry Leiba • Huawei Technologies

Internet standards development requires consideration of security issues in the protocols. But what does “security” mean in this context? We often conflate several different aspects into the blanket term “security.” Here, the author looks at some of these aspects separately.

As we develop standards at the various Internet layers, we must ensure that each standard, each protocol, is secure. We often talk about security with respect to computers and computer networks as though it were a clearly defined, monolithic concept. It’s not; security has several aspects, and, in differing contexts, we might refer to one aspect or another, or some varying combination. In particular, when we develop Internet standards, we often touch on these various aspects of Internet security.

I like to loosely split the general topic of security into the following subtopics:

- *Availability.* Is the system available when it’s needed?
- *Authentication.* Who am I, and how can I prove it?
- *Authorization.* What am I allowed to do?
- *Access control.* What data am I allowed to access, change, create, or delete?
- *Confidentiality.* Are communications and data safe from unauthorized viewing?
- *Integrity.* Are communications and data safe from unauthorized modification?

These aren’t absolute – you could certainly come up with a different set or choose to add to or remove things from the list, and some aspects overlap. Also, not all aspects apply to all situations. Many Internet services we use don’t need and wouldn’t benefit from authentication, require no access control, or present no confidentiality issues. And you’ll note that “encryption”

isn’t on this list – encryption isn’t security, but is rather a technology that can help establish aspects of security. We generally use encryption in authentication processes, for example, and to ensure confidentiality and integrity.

On the whole, it’s a good list to work from. As we design standards, protocols, and services, we must decide what aspects are important, and at what level of rigor we should apply them.

Availability

To provide context for these subtopics, I’ll be examining some of the threats Internet security mechanisms and standards try to defend against. One threat that came up in conversation recently was from an old *New York Times* editorial¹ about an investigation into overloading telephone lines for a political purpose:

[...] the New Hampshire phone jamming case was the real thing. Republican operatives hired an Idaho telemarketing firm to jam the lines to prevent people who needed help in voting from getting through. The scheme was a direct attack on American democracy.

The scheme was also what we call a *denial-of-service* (DoS) attack. In a DoS attack, the attacker demands so much service that legitimate users have little or no opportunity to get any. The one described in the editorial isn’t computer-related, but DoS attacks on websites are common, a popular way for a group to try to block a website that it doesn’t like.

We sometimes refer to distributed DoS – think about the difference between one phone

calling repeatedly with the redial button, as opposed to thousands of phones each calling (distributed) — but essentially every Internet DoS attack these days is distributed, and our defenses must assume that they are, so the distinction is mostly unimportant.

We can think of spam as a DoS attack as well: if your inbox fills with enough junk, it might be impossible to find the real mail. Worse, spam filters, designed to defend your inbox, might misclassify some mail as spam and delete it. Spam isn't generally meant to have this effect — something can become an unintentional DoS attack.

Defense against DoS attacks can be difficult because determining which service requests are legitimate can be problematic. Rate-limiting and block-listing are probably the most common mechanisms. Certain Internet addresses are known to be bad, and are blocked outright — all contact from them is discarded. Other addresses can make requests, but if they make too many in too short a time, they, too, are blocked, usually for some time period, although repeat offenders might be put on a permanent block-list.

Availability issues are considered in many Internet standards and related informational documents. For example, RFC 5782 addresses using block lists for spam, RFC 3882 is about preventing DoS attacks on the Border Gateway Protocol (a protocol for routing data on the Internet), and RFC 4732 looks at the general issue of denial of service on the Internet.

Authentication

Authentication is a precursor to some of the other aspects, for reasons that we'll see as we examine those further. It should be obvious, for instance, why authentication is related to authorization and access control. In particular, authentication mechanisms are built into many Internet-standard protocols. As we

update these protocols, we often seek to add new mechanisms that are more secure.

Everyone reading this is familiar with the authentication mechanism we started out with: some sort of user identifier (name, account number, serial number) and password. It served us well over the years, but isn't a very secure system, for several reasons. For one thing, people don't choose good passwords. If they're made to use good passwords, they record them in inappropriate places. Even what seem like good passwords often don't have enough unpredictability. And the password authentication systems themselves expose passwords to attack.

We can broadly divide what authentication mechanisms use into

and what you are (your signature). The latter combines what you have (the ATM card) and what you know (your PIN).

Another what-you-have mechanism is the SecurID device, which gives you a generated code that you can get only if you have the device with you.

Other what-you-are mechanisms use fingerprints, retina scans, and voice or handwriting analysis — collectively, biometric mechanisms. The most secure authentication systems combine multiple biometric mechanisms with an identification card and password, with all authentication information transferred securely. This makes a system that's pretty hard to break. Of course, it also makes one that can be pretty

People don't choose good passwords. If they're made to use good passwords, they record them in inappropriate places.

three categories: *what you know*, *what you have*, and *what you are*.

When you log into webmail, Flickr, MySpace, online banking, or online access to your credit-card account, the authentication mechanism you use employs what you know. Most what-you-know mechanisms are variations on the user ID/password combination, and all of them share the weaknesses I've described previously. The other mechanisms can help fix some of these deficiencies, especially when used in combination with passwords.

The most well-known combinations are point-of-sale credit-card purchases, where you sign the credit slip, and ATM transactions, where you enter a PIN. The former combines what you have (the credit card)

cumbersome to use. Biometrics are also subject to some serious limitations. If someone can spoof your left thumbprint, for example, you aren't really in a position to change it. And when you're ill, your voice-print might not be particularly useful.

Note, finally, that some people are reluctant to use systems that go beyond what you know, because carrying the what-you-have card or device is burdensome (what happens if you lose it or leave it at home when you're traveling?), and biometric readers can be expensive. But also, you might sometimes wish to let an assistant or some other delegate act on your behalf, and it's easy to give the delegate your password — but much harder to “lend” them your retina.

The answer to this is to understand the difference between impersonation and delegation, which goes beyond authentication and into the next two aspects, authorization and access control. The right way to handle delegation is to have the delegate authenticate with his or her own identity, and then be authorized to act on your behalf and receive access to the necessary information and resources. You should never allow another person to act on your behalf by impersonating you – there's no accountability in that.

Authorization and Access Control

I group these two together because they both deal with what the entity you authenticated as can do once you've logged in. I consider them separate aspects, however, because different mechanisms usually control each.

When I talk about authorization, I'm usually referring to actions that an authenticated user can take. Can you start and stop services, such as a Web server or a file transfer server? Can you shut the computer down? Can you add and remove users from a multiuser system? Can you send mail, install programs, change the system time, or set a computer's various other operational aspects?

Access control refers not to actions but to access to data. What files can you read? Can you create new files? What files can you modify or delete?

We'll collectively call what you're authorized to do and what access you're allowed *privileges*. Many computer systems, particularly those set up for use by more than one person, have two kinds of users: administrators and normal users. The former can do anything, and can get full access to all files. The latter are restricted in what they can do. On Windows systems prior to Vista, the lone user is generally set up as an administrator.

Those who try to do otherwise often run into difficulty because software (non-Windows software, that is – applications) assumes that the user's privileges aren't restricted. On MacOS, certain actions (such as updating the OS) and access to some files require that an administrator password be entered, essentially re-authenticating the user as an administrator. And for some things on MacOS, as on Linux, you must explicitly authenticate as the "root" user.

On the Internet, too, there are privileges. By logging into my Gmail account on my Web browser, I may send, read, and delete mail; manage my contacts; post to my blog and edit and delete blog posts; and send and receive instant messages. I can post comments to other blogs that use Blogger, and I can later delete those comments, but not other users' comments. On my own blog, I can delete anyone's comments, because I have that access. By using other authentication, I can access my credit cards, bank accounts, airline frequent-flyer programs, and so on.

Clearly, we must have restrictions on privileges over the Internet, but why should I want to limit my privileges on my own computer? Well, anyone who's made a mistake and deleted something accidentally, or gotten their computer infected with a virus while surfing the Web, should understand: if you don't have privileges that you don't need right now, you can't accidentally use those privileges to hurt yourself (well, to hurt your computer).

A rule of thumb called the least-privilege principle says that you should never be operating with more privileges than you need at the time. Most of us go around creating, modifying, and deleting personal files constantly, so we normally want such access. But how often do we need to delete files in the Windows directory, or in the System directory on MacOS? Seldom. And so we'd like

to avoid having that access unless we specifically ask for it.

And now we get back to something I said at the end of the authentication section: that authentication should be separate from authorization and access control. The right way to run a computer system is to have me authenticate as Barry, and then have privileges set up for what Barry can do and access. This provides auditability and accountability. If I want someone to be able to post to my blog and moderate comments in my absence, rather than giving him my Gmail password, allowing him to act as me in all ways (such as reading my mail, too), I should make sure he has his own blog account, and then give that account the privileges needed to manage my blog – but not my email.

Internet standards, too, often have delegation built into the protocols. For example, the Salted Challenge Response Authentication Mechanism (SCRAM; RFC 5802) allows for separate *authentication identity* and *authorization identity*, which allows delegation from the latter to the former.

Confidentiality and Integrity

Like authentication and access control, confidentiality and integrity are closely related: both deal with situations in which an attacker gets in the middle of the data stream. In the first case, the attacker is just snooping; in the second, the attacker is trying to modify or replace the data. These attacks are similar but have different characteristics and consequences. Note that I'm talking, here, about the confidentiality and integrity of data flowing through the system. Once the information is stored somewhere, a largely different set of threats and defenses are in play.

When you send a password, credit-card number, or other personal information over a computer network – and especially over an

open network such as the Internet – someone might be “listening in.” We think of information being sent from one computer to another, but it doesn’t happen quite that way. Networks are segmented to a significant degree, but at some level, your data goes out to a set of computers, with a specific computer’s address attached to it, and the other computers all ignore those data packets that aren’t addressed to them. Imagine if you received your postal mail by having the whole pile for your street left at the door of the first house, with each house’s occupant looking through the envelopes and keeping only those meant for him or her, then giving the rest of the pile to the next house. What happens on the Internet is something like this.

In this situation, someone could choose to keep a piece of mail meant for someone else, or could open one and read it before passing it on. The same is true with the Internet: a computer could be programmed to look at and record data intended for other systems.

The most common way to avoid this is to use data encryption, which can happen at the network layer, using IPsec (RFC 4301), on top of the transport layer, using TLS (RFC 5246) or SSL, or at the application layer, using standards such as S/MIME for email (RFC 5751).

When you visit a website whose URL begins with `https://`, your communication with that website is encrypted using TLS or SSL. The Web browser ensures that the computer you’re talking to has security credentials that match the address in the URL, then negotiates encrypted communication. A computer program can still peek at and record the data packets – but a snooper won’t be able to decipher the data, which will thus be useless. Similarly, if a snooper should try to replace or modify the data you’re sending – say, to change a \$20 payment to \$2,000 – encrypted

communication would prevent the attacker from being able to modify the encrypted information in a valid way.

Encrypting an entire communication, however, has been fairly expensive in the past, slowing down the communication. Encrypting the information you get from Wikipedia or the *New York Times* is fairly unnecessary, so to speed things up, we don’t encrypt everything on the Internet. This is, however, changing, as computing speeds have increased to the point where Web traffic encryption is no longer a performance issue.

Sometimes, though, it’s not important to protect information from prying eyes, and the likelihood of its being altered by an attacker is small – but it’s important enough that we want to know if it’s been altered. In such cases, we don’t need to prevent the alteration, but we do need to detect it. For that, we can use digital signatures.

A detailed explanation of digital signatures goes beyond this column’s scope. The short version is that they provide a mechanism for ensuring that the person we think sent information is actually the person who sent it, and that it wasn’t altered along the way. Otherwise, we know something is wrong – we don’t know how to correct it, but we know to ignore the faulty data.

Of course, all discussion of encryption and digital signatures here assumes that the encryption technology and algorithms used are current and sufficiently strong, and are used properly. This is usually the case, but weak and compromised algorithms are still used on the Internet surprisingly often. As developers of Internet standards, we often update the standards to deprecate the older algorithms and replace them with stronger ones. Still, it takes time for deployed software to catch up. As a user, your best defense is to make sure you’re using

a current Web browser (and other software, such as mobile apps), and that you’re keeping the browser and the operating system updated regularly. Current versions of Firefox, Internet Explorer, and Chrome no longer support older, flawed versions of SSL, or they have those old versions disabled by default. So stay up to date.

Standards development organizations have shown increasing awareness of the need to think about security at every stage of development, and to consider what aspects are needed for the protocols and use cases they’re developing. The IETF, for example, has an organizational area devoted to security, and every document must have a Security Considerations section that describes what the issues are for that document. ATIS has started a focus group on cybersecurity. And, of course, the IEEE’s Standards Association includes security review for appropriate standards. □

Reference

1. “The New Hampshire Phone Scam,” *New York Times*, 17 Sept. 2007; www.nytimes.com/2007/09/17/opinion/17mon3.html.

Barry Leiba is a standards manager at Huawei Technologies. He currently focuses on the Internet of Things, messaging and collaboration on mobile platforms, security and privacy of Internet applications, and Internet standards development and deployment. Leiba has been active in the IETF for roughly 15 years, is an author of several current and pending proposed standards, has chaired numerous working groups, served on the Internet Architecture Board from 2007 to 2009, and is currently serving as Applications Area Director. He edits this column, and can be reached at barryleiba@computer.org.

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.