

Security 2

Almost-SSL Example

This is almost how SSL works...

Alice contacts Bob

Bob forms pub/priv at random, sends pub to Alice

Alice forms "session" key k , sends to Bob, encrypted with pub

Now both have session key k , for traditional encryption for the duration of the session

Attack: Man In The Middle (MITM)

Carl takes over router between Alice and Bob

Alice thinks she is talking to Bob, but in fact is talking to Carl

Bob thinks he is talking to Alice, but in fact is talking to Carl

For the Almost-SSL example above, Carl can substitute his own pub/priv talking to Alice, and so get k and observe or alter the communication

MITM Truism

If two parties do not share any prior info, they are always subject to the MITM attack

Anything Bob can do, Carl can do

If Alice does not have any prior knowledge about Bob, there is nothing Bob can do to distinguish himself from Carl

Attack: DNS poisoning

A way that Carl might get between Alice and Bob

Feed bad data to the local DNS server, so that when a local user contacts "www.bank.com", the DNS server gives Carl's IP addr.

Carl puts up a page that looks like the bank page

Attack: Fake ATM

Over long weekend, put up a fake ATM in front of the real ATM, with some "work in progress" signs.

People put in their card, type in their PIN number

Carl records the info, spits out the card with an error message

Later, use the info to pillage the accounts

Conclusion: would like mutual authentication -- each party can verify the identity of the other

MITM Solution - Certificates

Alice contacts Bob

Bob sends back certificate (cert) giving Bob.pub key

The Cert is signed by a Certificate Authority (CA), known to Alice.

The Cert associates Bob's identity/DNS name, with Bob.pub key

Alice verifies that the cert is valid, since the cert is signed by the CA

Alice forms session key k , sends out with Bob.pub

Carl can intercept the traffic, but cannot do anything with it. Only the true Bob can recover k , since only Bob knows Bob.priv.

The CA acts as a 3rd party, verifying for Alice what the pub key is for the alleged other party.

Own CA

You can be your own CA if you like, you just need to add yourself to the local list of trusted CAs used by your browser or email client.

e.g. foo.com sets up three servers, A, B, C, each with a pub/priv, signed by the foo.com CA key. The programs on A, B, C are set to trust the foo.com CA

When A connects to B, it gets and verifies the cert as usual -- protecting against MITM

MITM Counter-Argument

There is an argument that the MITM attack is, in practice, extremely rare.

HTTPS uses CAs to provide both security and authenticity, proof against an MITM attack

The whole CA system is complex (and somewhat expensive) because it protects against MITM

There is an argument that when, say, I'm using a web shopping cart, what I really want is just security. The odds of someone doing MITM are unlikely., and therefore it is unfortunate that heavy HTTPS/CA design insists on solving both problems.

Identity Conclusions

Don't trust DNS/IP addr to provide identity of other party

Ways to auth the other party...

1. Shared secret key, exchanged earlier
2. Cert identifying the pub key of the other party. Send them a session key (any token will do) under that pub key -- only the correct other party will be able to decrypt and use the session key

Firewall

Similar to NAT

Separates the internal net from the rest of the internet.

Only allow certain port#'s connections to go through in certain directions.

e.g. incoming port 80 requests, only allowed to the web server, not any other internal IP addr.

Also, can provide a false sense of security -- if the HTTP server has a vulnerability, the firewall is no help.

Modern worms, traverse in multiple ways (email), so they get inside the firewall
Simple firewalls just work by port# and direction

More complex firewalls understand the protocol (HTTP) and can block/kill connections based on more specific criteria (e.g. block HTTP connections to a certain site)

Firewalls can block legitimate network usage -- creates a tension where for the security people, it's easiest to just block all but a handful of ports. As a result, many protocols use port 80 and HTTP or HTTP-like protocols, since that is sure to go through the firewall.

Virtual Private Network

Set up an encryption layer in software between all your computers on the Internet. The VPN traffic is all encrypted/authenticated before being sent inside regular Internet packets on the Internet at large.

IPv6 does this

Security -- Fear Culture

There are many, real security problems on the net today
However, the security tool vendors and some "experts" will naturally tend to exaggerate the problems.

My policy: make sure your browser and emailer have all their most recent patches. Consider not running Microsoft client software -- they have tended to have the most problems for a variety of reasons. Be careful with email attachments. That's all I do, and I don't run any special security software, and I have not had a problem in over 10 years.

Security -- Blame The User

Many computer users are casual, and do not actively maintain their system
Blaming the user for not being up to date is not a realistic way to deal with security.

A well engineered solution will not depend on the user to be completely on the ball.

Server Security vs. Complexity

Set up server with web server, file sharing,

The system is complex, so it's hard to get every detail right.

If there are 1000 details, it takes real dedication to get all 1000 right.

Security Optimism

The technology for doing security right (better languages, better protocols, ...) is getting better.

I believe that security problems will be less of an issue in the future -- we are currently experiencing a vulnerable sort of adolescence that will get better.