Cryptography and Network Security

PROFESSIONAL EDUCATION

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Network Security and Protocol Design

David Clark

Senior Research Scientist

Computer Science and Artificial Intelligence Laboratory (CSAIL)

Massachusetts Institute of Technology

PROFESSIONAL EDUCATION

CSAIL
MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY

Introduction—Scope of the topic

These presentations are about network security.

- I will focus on the Internet.

I will address three questions:

- What are the different security problems we must consider?
- What are the methods in use to deal with these problems?
- Why does Internet security seem so poor?
  - This last question may be the most important.

Some key considerations

The Internet is a global network.

- Different parts belong to different actors.
- Not all are equally trustworthy.

There are billions of users.

- Not all of them are equally trustworthy.

We cannot eject the untrustworthy elements.

- Just have to make sure they cannot harm us.

My list of security problems

Trusting users try to communicate

- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

One user attacks another.

- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.

- One part attacks another.
- Users attack the network.

Denial of service attacks.

I will deal with each of these in subsequent presentations.

My list of security problems

Trusting users try to communicate
- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

Two parts:
- Confidentiality and integrity
- ~~Availability~~

One user attacks another.
- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.
- One part attacks another.
- Users attack the network.

Denial of service attacks.

I will deal with each of these in subsequent presentations.

Attacks on communication

Use the classic security sub-goals:

- Confidentiality
- Integrity
- Availability

Cryptography is a powerful tool

- Encrypt content -> confidentiality protected.
- Signed content -> loss of integrity detected.

Encrypted communication is used in several ways in the Internet.

Uses of encryption in the Internet

IPsec:

- Protects packets at the IP layer.
- Typical use:
  - VPN tunnel.

Transport Layer Security (TLS):

- Protects a stream of bytes.
- Typical uses:
  - Protected connections to Web sites.
  - Protected connections to mail servers.

Quick tutorial on TLS

Example: client connecting to secure server.

- "https" connection to web server.

Exchange of packets at the start do the following:

- Allow the client and server to negotiate which crypto scheme to use.
- Allow the client to verify identity of server.
- Use the public-private key pair of the server to establish a shared key to encrypt the rest of the conversation.

Initial TLS exchange

C -> S:

- Here are some crypto options.
- Here is a random number R1.

S -> C:

- Here is the crypto scheme I pick.
- Here is a random number R2.
- Here is a certificate saying who I am (includes public key).

Client should verify the certificate.

C -> S:

- Here is a random number R3 encrypted using your public key.

C & S each use R1, R2, R3 to compute session key.

- Start encryption using that key.

What is a certificate?

A certificate contains (to simplify):

- The DNS name of the server.
- The name of the organization.
- The public key of the organization.
- A trusted third party that vouches for this information.
  - Cryptographically signs it.
  - These trusted third parties are called *certificate authorities (CAs).*

But how does the client know to trust the third party?

- Who vouches for the third party?

Trust hierarchy

A CA can vouch for the server.

A “higher-layer” CA can vouch for that first third party.

- And so on.

But there has to be a shared agreement about the “highest-level” CAs for this scheme to work.

- Roots of trust.

How does a client today know about the top-level CAs?

- The list comes pre-loaded in the browser.
- The real root of trust is the distributor of the browser
  - Mozilla, Apple, Microsoft, Google

Two problems with this scheme.

What if a CA is actually not trustworthy?

- Issues false certificates?
  - CA could be corrupt, penetrated, or adversary.
- Does this happen in practice? YES!
  - Dutch CA DigiNotar was penetrated, apparently by Iran.
  - Google recently declared China CA untrustworthy after false certificates were used in an attack on TLS-protected communication.

What about availability?

- The next presentation on this topic.

Availability—the third dimension of security

Confidentiality and integrity were (sort of) addressed using cryptography.

- But with an odd side-effect.

A failure of integrity (e.g., malicious modification of the data) terminates the connection.

- An attack on integrity becomes a successful attack on availability.

Attacks by untrustworthy nodes in net

Does this integrity attack matter?

- An untrustworthy node could just drop the packets, if the goal is loss of availability.

Users “click through” warnings about untrustworthy keys because they want to make progress.

- Security experts worry about confidentiality and integrity, users care about availability.

Availability—a general theory

Hard in general to coerce an untrustworthy actor into working properly.

- In particular, not with technology.

To enhance availability, the system must:

- Identify a failure or attack when it occurs.
- Localize this event to a region or component.
- Reconfigure communication to avoid that region.

Availability—localize and reconfigure

This framework is well-understood with respect to failures:

- Links and routers fail; the Internet has dynamic routing.

Not so obvious with respect to attacks:

- Which part of the system can detect attacks?
  - The end-points.
- Which part of the system can reconfigure routing?
  - Not the end-points.

Today, imperfect solutions involving humans.

On to the next topic

Trusting users try to communicate

- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

One user attacks another.

- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.

- One part attacks another.
- Users attack the network.

Denial of service attacks.

Untrustworthy users

The previous problem:

- Users trying to communicate are attacked by network element.
- Implication: they had interests in common and were mutually trusting.

The reality of today:

- Most communication on the Internet is among parties that do not trust each other and with good reason.
  - Email: spam, phishing, malicious attachments.
  - Web: forged web sites, downloaded malware, profiling.
- But on balance, users proceed.

When trust is lacking…

When trust is lacking, society falls back on constraints.

- Escrow, deed registries, notaries, credit cards, etc.

End-to-end encryption may be the whole wrong idea.

- A private channel between you and your attacker?
- Consider physical mail: private communication or anthrax in that envelope?

How is constrained communication realized?

- By the design of applications, not the network.

The network and the application

The network, by design, is general.

- It does not know what the users are trying to do.
- It just moves sequences of packets.

Applications, by design, define the actual flows of data.

- Applications define the experience of using the network.

The network *should not* know what the users are trying to do.

- Would make it easier for net to attack users.
- Might raise barriers to deployment of apps.

Protecting the host—then and now

Old thinking (early design era):

- The host must protect itself, because it cannot trust the network.
- The network must protect itself, because it cannot trust the host.
  - Early advice from NSA…

New thinking (based on reality):

- The host cannot protect itself.
  - Over-large software base.
  - Applications that are insecure by design.
- Hosts could do a better job.
  - Sandboxing, for example

Consequences of new thinking

Cannot trust “the net”, but perhaps trust a region of the net.

- Example: trust my region of the net not to route around my firewall.

Use encryption to enforce patterns of communication among elements that are deemed trustworthy.

- TLS connection to mail server.
- Protects from some attacks by network
  - Hot spots, repressive nations, etc.
- Can perhaps help localize points of attack.

Applications: insecure by design?

Users favor features over constraints.

- Sending arbitrary attachments in email, downloading Javascript, etc. is very useful.
- Makes good sense when parties are prepared to trust each other.
- Users favor both features and availability over potential security concerns.

Possible design approach for apps:

- Vary feature set depending on degree of trust.
- “Don’t accept Javascript or attachments from strangers”.

Trust-based variation in function

For apps to modulate their behavior based on the degree of trust:

- They have to know who to trust.
- They have to know the identity of the participants.

Implication: identity management is key to trust-based protection.

- Identity theft is deeply corrosive to good security.

So should the network impose an identity architecture?

My view: no!

- Needs for identity vary by context.
- Apps define the context.

What if the host is compromised?

If a malicious actor has taken over a host:

- All the roles reverse.
  - It may attempt to open private communications to its partner "bad guys".
  - We try to block them.
  - (Just like repressive governments…)

What can mitigate this situation?

- Again: application design.
- Design apps that depend on multiple components and can detect that one component has failed.
- (Suggests a new role for net: blocking unauthorized flows.)

Depending on the apps for security

If you accept my logic, I have taken us to a nightmare place.

- Instead of “the network” protecting us once and for all.
- Each application has to take on the role of protecting us:
  - Modulate dangerous features based on trust.
  - Implement identity schemes adequate to track trust.
  - Implement communication patterns that protect us from attacks by the network.
  - Integrate multiple elements that can detect of one element is flawed.

But…it is applications that translate data flows into actions.

- It is actions that cause harm.
- Actions occur in the end-nodes.
- If the end-node cannot protect us (sandboxing?) the app must.

Protecting the insecure host

We must accept that a general purpose operating system will have flaws.

- But it is reasonable to imagine simpler components that are much more secure.

An application can both protect the host and itself become more secure by moving critical functions to such a “helper” device.

- Pre-processing of the data stream.
- Identity validation.
- Granting approval for outbound data flows.

This approach only works if this structure is integrated into the workflow by the application.

My list of security problems

Trusting users try to communicate

- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

One user attacks another.

- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.

- One part attacks another.
- Users attack the network.

Denial of service attacks.

One part of the network attacks another

Look at one example: global routing algorithms.

- Many examples today of one part of the network making false routing assertions to mess with global routing.

The “obvious” solution was proposed a while ago:

- Have regions of the network use cryptography to sign their routing assertions so they can be validated.

The flaw (just as with TLS): who is trusted to vouch for all the regions of the Internet.

- One global root of trust? Not workable in real world.
  - If I have the power to vouch for you, I can “unvouch” you.
- Multiple roots of trust? Some turn out not to be trustworthy.

A novel approach to certificates

Forget about certificates and trusted third parties (the CAs).

Anyone can make up a public-private key pair and sign anything.

- The first time you get a signed assertion, you have no idea whether to believe it.
  - Just like today, with no signatures.
- But over time, receivers learn which signatures are trustworthy.

Not technically robust, more socially robust.

- Key continuity: a “getting to know you” protocol.

My list of security problems

Trusting users try to communicate

- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

One user attacks another.

- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.

- One part attacks another.
- Users attack the network.

Denial of service attacks.

Denial service attacks

Use many computers to send unwanted traffic toward a host or region of the network, thus overwhelming it.

- Often called Distributed Denial of Service attacks (DDoS), because of the many computers involved in the attack.
- Fits in several of the previous categories, but different in character.

Where did all those computers come from?

A malicious actor (a “bot-master”):

- Penetrates lots of computers (see previous presentation);
- Installs malware (makes them into “bots”);
- Hooks them into a common control structure (makes them into a “botnet”);
- Sells time on his botnet to other bad guys.

Botnets can be used for many purposes:

- DDoS attacks
- Sending spam

Dealing with botnets

Several options, all tricky:

- Make it harder to penetrate hosts (see above).
- Remove malware from bot.
  - Who has the ability and the authority? An incentive issue.
- Disrupt control structure (see above—trusted communication).
- Mitigate harmful behavior. Different for different harms:
  - Spam: block high-volume outgoing email.
  - DDoS: several approaches, all tricky.

Mitigating DDoS attacks

Today:

- Dissipate the attack against lots of replicated targets.
  - Commercial firms offer this service today.
- Block certain forms of attack.
  - Source address verification would prune some attacks.

Tomorrow (maybe):

- Flag traffic as suspect and discard at locus of attack.
  - But who gets this power? Would this be a new attack vector?
- Receiver tells senders to shut up.

Conclusions

To wrap up, some summary thoughts.

Remember my starting slide

I proposed to address three questions:

- What are the different security problems we must consider?
- What are the methods in use to deal with these problems?
- Why does Internet security seem so poor?
  - This last question may be the most important.

My list of security problems

Trusting users try to communicate

- Untrustworthy elements attack the communication.
- Could be a network operator, for example.

One user attacks another.

- As part of intentional communication.
- Without any desire to communicate.

The network itself is attacked.

- One part attacks another.
- Users attack the network.

Denial of service attacks.

Summary thoughts--I

Under different circumstances, no part of the system is trustworthy.

- Sometimes a region of the net is not trustworthy.
  - Consider the extreme case—repressive governments.
- Sometimes the end-nodes and users are not trustworthy.
- Sometimes the end-node software is not trustworthy.
- Sometimes the trusted third parties are not trustworthy.

Summary thoughts--II

The overall design must somehow accommodate this variation in trust.

- If *nothing* is trustworthy (think repressive government), there may be little that can be done.
  - Situation deteriorates to a cat and mouse game played against security forces. Not a good game in the limit.
- In most cases, there are trustworthy components—the system must allow the user to identify and exploit these.
  - Assuming that the goal is to let the user have those choices.
  - The actor that controls choice controls the power.

Summary thoughts--III

Cryptography is a powerful tool:

- One can construct technical arguments about its strength.
- It can be exploited in all sorts of clever ways.

Trust management is the ugly duckling:

- Cannot “prove” things.
- No technical arguments about strength.
- Defined by social context, not technical issues.

It is unfortunate that almost all uses of cryptography sit in a larger context of trust management.

- Old security saying: amateurs attack the algorithm, professional attack the key management scheme.
  - Trust management is the bigger ugly duckling sibling of key management.

Summary thoughts--IV

The regrettable role of applications.

- It would be wonderful if the network layer could "solve" the "security problem" independent of the application.
- But it can't.

What the lower layers of the network and the end-node must do is make it easier to design secure apps.

- Several of the security problems I have described can be mitigated through application design.

Summary thoughts--V

I divided the landscape of security along system boundaries:

- What part of the system was being attacked.

There is another way of mapping the landscape of security:

- Who got harmed?
- One of the problems with system penetration is that the event does not signal who will eventually be harmed.
- With many harms, our tendency is to punish the defender.
  - Think data breach and credit card fraud.

A harms-based approach reminds us that the defense may be spread across many actors.

- A serious coordination problem and an invitation to play “security nimby”.

Summary thoughts--VI

My views on network security may well not be shared.

I believe my logic, but the security field is shaped by persistent disagreement.

Failure to work toward agreement is a great recipe for not making progress toward better security.

What sort of leadership would be accepted?

- Globally? (Just to make the problem harder).

THANK YOU

David Clark

Senior Research Scientist