Progress

~~Introduction~~

~~Software security~~

~~Host security~~

Network security

Web security

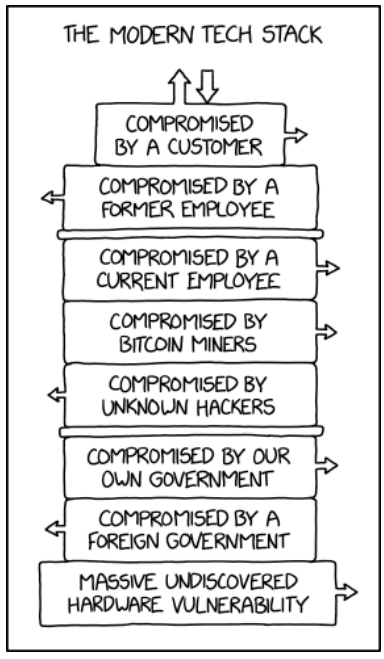
Today

Networking layers

Networking assumptions

Networking attackers

Recall


THE MODERN TECH STACK
COMPROMISED BY A CUSTOMER
COMPROMISED BY A FORMER EMPLOYEE
COMPROMISED BY A CURRENT EMPLOYEE
COMPROMISED BY BITCOIN MINERS
COMPROMISED BY UNKNOWN HACKERS
COMPROMISED BY OUR OWN GOVERNMENT
COMPROMISED BY A FOREIGN GOVERNMENT
MASSIVE UNDISCOVERED HARDWARE VULNERABILITY

Common layers / encapsulations

#	Name	Example
7	Application	HTTP, DNS, NFS, SSH...
6	Presentation	TLS, SSH...
5	Session	SOCKS, SMB...
4	Transport	TCP, UDP, SCTP...
3	Network	IP
2	Data link	Ethernet MAC
1	Physical	Ethernet PHY

OSI model in practice

OK, that's kind of a nice idea, but...

- rigorous standardized layers most important for lower layers
- fairly narrow-minded (e.g., encryption goes **there**)
- can be a crutch to avoid thinking about layering for yourself

Still: useful to think about encapsulation, abstraction layers, TCBs

Standardization matters a lot at some layers of the stack. However, _____ _____ is a bit more fluid at the top of the stack than at the bottom.

Let's not be dogmatic about where the crypto should go, or which layer is supposed to _____. We need to think _____ about these things!

Starting a section of a technical report with, "the OSI model for networking states that..." is like starting a speech with, "_____." It's a terrible way to engage both your own mind and your reader's: instead of _____ first, it _____ and then _____ _____.

_____ show Wireshark example

TCBs in networking

Q: What is the TCB in networking?

A: It's complicated!

It depends on:

- how you trust middleboxes
- how you trust the other host
- how you trust other users

Different networking models embody different assumptions about trust.

We'll talk more about middleboxes as we get further into the course, but for now, we should recognize that different networking arrangements place different levels of trust in the switches, routers, caches and TLS interception boxes that exist between you and a remote host.

When we talk about "trusting" a remote host, we should (as always!) think carefully about what exactly we mean. We can trust a host to have our best interests at heart (unlikely?), we can trust a host to deliver something they said they would (more likely), we can trust a follow a protocol correctly even if they're malicious, etc.

Assumptions

Original internetworking model

Trust

- physical security
- low-numbered ports
- raw sockets

Systems tend to embody the assumptions of their designers. This observation has been expressed in different ways, including Conway's Law: teams tend to design software whose structure mirrors that of their organization. Teams in hierarchical organizations think in terms of hierarchy, teams in flat organizations think in terms of interconnection webs, etc. Similarly, things like network protocols embody their designer's assumptions about "how things ought to work".

Early internetworking was done by a handful of sysadmins at a handful of institutions (mostly universities), most of whom knew each other and all of whom had control over the systems attached to their networks. How would this affect the internetwork they designed?

Trusted networking

Suppose we chose to trust everything:

How could you attack an application, e.g., online banking?

What *must* we trust?

Could we eliminate trust in any or even **all** of these layers?

- application data
- remote hosts
- local OS
- internetworking middleboxes
- network links
- physical media

Eliminating trust

Answer: no. We can't eliminate trust in *all* layers, as you ultimately must choose to trust something or someone in order to get anything done. In the online banking case, you've chosen to trust the bank (backed by a regulator, backed by a government)... if you don't like trusting banks, try putting your trust in a global cabal of cryptocurrency developers!

How little can we trust?

(a slightly depressing question!)

Cryptography: Kirckhoffs's principle (one of six)

> *It should not require secrecy, and it should not be a problem if it falls into enemy hands*

Networking: the *Dolev-Yao* attacker

The Dolev-Yao attacker*

Communications should assume an attacker can:

1. observe all messages (**passive eavesdropper**),
2. send messages impersonating users (**active** attacker) and
3. intercept messages and drop or replace them (**middleperson**).

* Dolev and Yao, "On the Security of Public Key Protocols", *IEEE Transactions on Information Theory* 29(2), 1983. DOI: 10.1109/TIT.1983.1056650

Like with Kirckhoffs's principles, assuming a very strong attacker will help us defend against both strong and weak attackers. Assuming a weak attacker will lead to designing vulnerable systems that can't stand up against strong attackers.

Dolev-Yao in practice

What would these assumptions mean for the TCB?

This is an example of an *end-to-end* argument*: what matters is the end-to-end communication, the middle is just detail. (spidey-sense?)

How can we put these assumptions into practice?

Trust no one! Or...

Be explicit about trust and communication: **security protocols**

* Saltzer, Reed and Clark, "End-to-end arguments in system design", *ACM Transactions on Computer Systems (TOCS)* 2(4), 1984. DOI: 10.1145/357401.357402

Do your spidey-senses tingle when I say that something is "just detail"? They should! That is _____.

Summary

Networking layers

Networking assumptions

Networking attackers