Discussion

- End-to-end security
- Robustness principle
- Validating input data
- Writing secure code

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

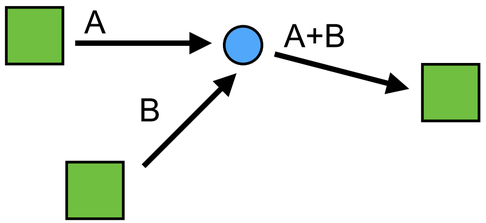
End-to-end Security? (1/2)

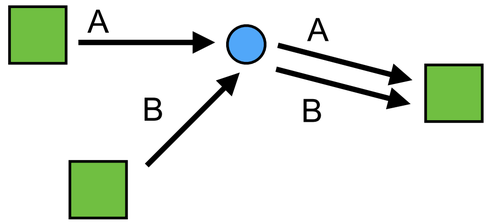
- For communication to be secure, it must be **end-to-end**
- The two endpoints are the sender and the final recipient
  - With a data centre or CDN, what is the final recipient?
    - Is it the load balancer at the entrance to the data centre, or the server within the data centre that processes the request?
    - If the request is directed to a content distribution network (CDN), is the end the local cache that serves the request? If so, how are the encryption keys shared?
  - If the data is moving between two users, is it encrypted between the two users or between each user and the data centre?
    - i.e., can the data centre see user-to-user data flows?
    - (This is normal if you run TLS from user-to-data centre then from data centre-to-user)

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

End-to-end Security? (2/2)

- Is there in-network processing? How much data is revealed to the in-network server?
  - e.g., video conference with privacy protection vs. without
  - Does the central server decrypt the speech data, mix into one stream and send to the receiver, or does it forward all active streams in encrypted form





  - Trades-off security vs bandwidth
    - For audio the bandwidth is small enough this doesn't matter
    - For video conferencing, the combined bandwidth may be significant

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

The Robustness Principle (Postel's Law)

> At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability:
>
> "Be liberal in what you accept, and conservative in what you send"
>
> Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by un-envisaged mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

RFC1122

Balance interoperability with security – don't be *too* liberal in what you accept; a clear specification of how and when you will fail might be more appropriate

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

The Robustness Principle (Postel's Law)

"Be liberal in what you accept, and conservative in what you send"

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

The Robustness Principle (Postel's Law)

"Be liberal in what you accept, and
conservative in what you send"

**"Postel lived on a network with all his friends.
We live on a network with all our enemies.
Postel was wrong for todays internet."**

Poul-Henning Kamp

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

The Robustness Principle (Postel's Law)

Network Working Group — M. Thomson
Internet-Draft — Mozilla
Intended status: Informational — November 04, 2019
Expires: May 7, 2020

The Harmful Consequences of the Robustness Principle
draft-iab-protocol-maintenance-04

Abstract

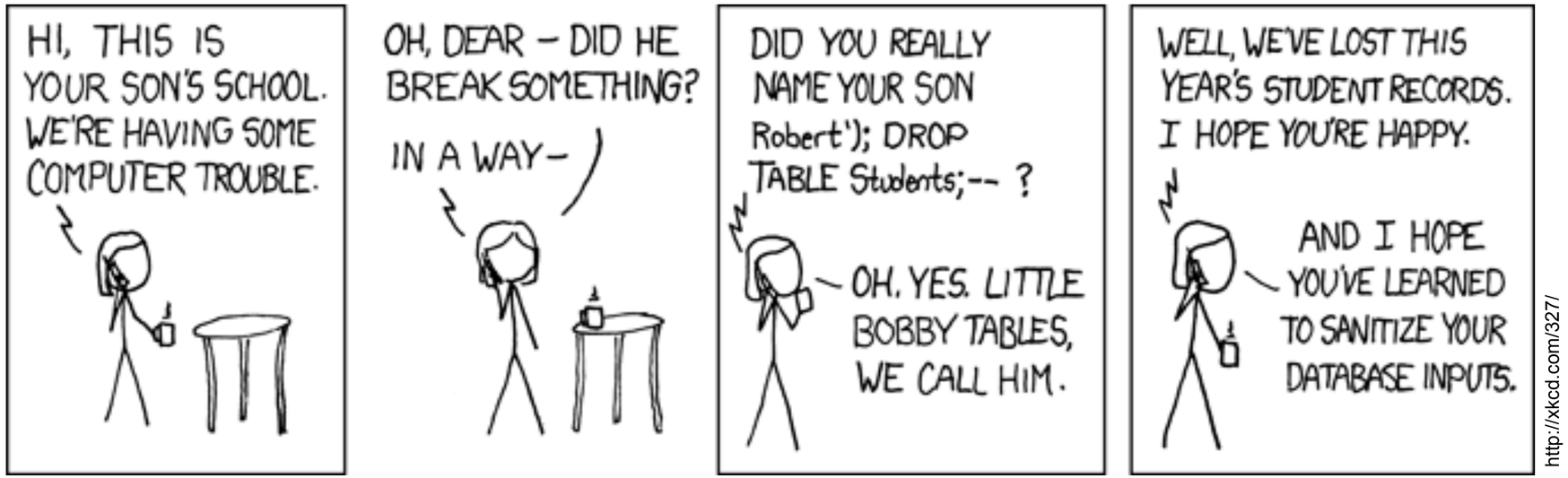
The robustness principle, often phrased as "be conservative in what you send, and liberal in what you accept", has long guided the design and implementation of Internet protocols. The posture this statement advocates promotes interoperability in the short term, but can negatively affect the protocol ecosystem over time. For a protocol that is actively maintained, the robustness principle can, and should, be avoided.

Note to Readers

https://tools.ietf.org/html/draft-iab-protocol-maintenance-04

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Validating Input Data



http://xkcd.com/327/

- Networked applications work with data supplied by un-trusted third parties
- Data read from the network may not conform to the protocol specification
- Due to ignorance, bugs, malice, or a desire to disrupt services
- **Must carefully validate all data before use**

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Writing Secure Code

- The network is hostile: any networked application is security critical
  - Must carefully specify behaviour with both correct and incorrect inputs
  - Must carefully validate inputs and handle errors
  - Must take additional care if using type- and memory-unsafe languages, such as C and C++, since these have additional failure modes
- **The best encryption doesn't help if the endpoints can be compromised**

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow

Secure Communications

- The need for secure communication
- Principles of secure communication
- TLS v1.3
- Discussion

Colin Perkins | https://csperkins.org/ | Copyright © 2020 University of Glasgow