Utu: Saving The Internet With Hate

The Grand Experiment

- An experiment in sociology and computing

Philosophical Foundations

- Strong Identity
- Reputation
- Retribution

Identity

- Nobody knows how you are on the internet.

Reputation

- But, they know you're a dickhead.

Retribution

- And, wouldn't it be great to punish the dickheads?

Technical Solutions to Social Problems

- If it goes on the Interwebs, it's social.

The Protocol

- Sends 2 bytes for size (htons(size))
- Sends the data structure
- That's it.

Framing, Lexemes, Grammar

- Uses stackish, an XML joke.
- Google for it.
- S-expressions with netstrings in a FORTH stack order.
- [['4:test' 1234 child root
- (root (child (1234 '4:test')))

Semantics

- Determined by the Hub.
- One Stackish node for header.
- Another for body.

Data Encoding

- Always ASCII text, but you can put anything you want in BLOBs.

Simplicity Is Key

- 2 bytes for frame
- Stackish for header and body
- Easy to implement

Security Preventions

- Client does all the work first.
- Client is considered hostile.
- Booted immediately.
- A Finite State Machine controls.

The Cryptography

I'M NOT A CRYPTOGRAPHER

- I didn't write any crypto
- Followed the rules

Standards Used

- AES 32-byte (256 bits)
- ECC 32-byte (256 bits)
- SHA256 hashes
- 128 bit random nonces
- ISO/IEC 11770-3 Mechanism 6 without the Helsinki vulnerability
- CCM based encryption with AAD
- Fortuna PRNG

Implementations

- utuprotocol.info for server
- ihate.rubyforge.org for client

Secure Coding Practices

Valgrind, Valgrind, Valgrind

- All code is ruthlessly run through it.
- Uh, except you can't run ruby through it.
- Sigh, ruby sucks.

Statistical Quality Control

- Collect information on
  - Valgrind errors
  - Unit test errors
  - Logged Errors
- Analyze the change over time

Code Auditing

- Review my code repeatedly.
- Consistent “rules” I follow
- Assume it's always broken

C Coding Practices

- Bstring library for strings
- All functions have validations
  - Assert
  - preconditions/postconditions
  - Extensive unit testing

Secure Unix Server Practices

- Always chroot. ALWAYS.
- Leave minimum resources open.
- Minimum configuration.

Practical Demonstration

Current Security Flaws

- Messages are numbered sequentially
- No versioning on packets
- Cryptography Not Evaluated

Feedback and Questions

Call For Assistance

- Entirely GPLv3 Code
- Open and contributions welcome
- savingtheinternetwithhate.com
- ihate.rubyforge.org