# Dagstuhl Workshop Presentation: Learnings from Failure

Geoff Huston, APNIC

# The 40 year Scorecard for the Internet

- ❑Stacked Protocol Architecture
- ❑Stateless Packet Switching
- ❑ACK-paced Transport
- ❑End-to-End
- ❑Network Management
- ❑Names and Referential Frameworks
- ❑Security

# The 40 year Scorecard for the Internet

None of these areas are unqualified “successes” for the Internet Protocol:

- Layer violations proliferate
- MPLS is a widely used circuit-based overlay for many IP service platforms
- Ack-paced transport is slow and clumsy
- The proliferation of network middleware shows a moderated view of “pure” end-to-end
- The relationship between networks and hosts has never been made to work
  (The saga of failed QoS efforts, for example)
- UDP is a glaring vulnerability in the DOS world

# Learning from our Failures

IPv6

- What happened?
- Why didn’t the industry take a low risk path and avoid IPv4 address depletion completely by adopting IPv6 earlier?
- Is IPv6 too complex for a simple-minded industry?
  - Fragmentation and Extension headers
  - SLAAC / DHCP / etc
  - Chaotic address plan
- Is the lack of backwards compatibility a fatal flaw in any transition?
- Do we understand the economics of transition in a heterogenous market-driven environment?

# Learning from our Failures

DNSSEC

- Why doesn’t anyone validate what they learn from the DNS?
- Why don’t zone admins DNSSEC-sign everything?
- The DNSSEC design parameters looked good:
  - Backward compatible, incremental extension, no protocol change to the DNS
- But its just not happening
  - Resistance to signing
  - Resistance to validating at the edge
- Is this a case of technical failure or economic failure? (or both!)

# Learning from our Failures

## Security and Robustness

- Obviously what we are doing is nowhere near enough - we are engaging in a spiral process of escalation of the sophistication of the attacks and a response of greater complexity
- Some commentary has suggested we would have been in a better place if we had ”integrated” security into the basic protocol design at the outset
- Other commentary suggests that we really never knew how to do that and a bolt-on approach allows for greater agility in our response
- Have we largely given up on network level security and just relying on applications to fend for themselves?
- Is this enough?

# A Case Study – Routing Security

- The Internet took the novel approach of self-orchestration in building the routing system
- Routing is a distributed algorithm operating across a collection of diverse mutually trusting entities
- Routing is easily abused:
  - Deliberately by manipulation of the routing exchange
  - Deliberately by a rouge entity abusing the trust model
  - Inadvertently through operational mishap
- We assume(d) that if we addressed a packet to the “right” IP address, then we could trust any response we received as “authentic”
  - Corrupting the routing system undermined this assumption

# Routing Security is a Hard Problem

The problem space includes:

- The integrity of the routing information
- The proper operation of the routing protocol to propagate this information
- A definition of points of “absolute trust”

It’s taken more than 30 years and we’ve achieved little

# RPKI and BGPSEC Efforts

The RPKI is a poor fit to the routing application space

- Routing needs to bring “all” of the PKI to every relying party – this is a scaling flooding problem
- Routing is not transactional – the information in a routing update is trusted until it is withdrawn or updated
- Omissions (deliberate or otherwise) in routing propagation are unverifiable
- Routing security works best when everybody signs and everybody validates

BGPSEC is over-burdened

- AS path validation is implemented by signing chains which impose a high crypto burden on routers - BGP session resets become a problem
- Loading keys into routers and maintaining this framework is operationally brittle

# Does anyone care any more?

- After 30 years with no workable secure routing framework applications have resigned themselves not to trust the underlying network
- TLS is the general “solution” to this problem
  - Irrespective of where the packet may have been delivered, if the remote end cannot demonstrate that it is a genuine instance of the named service then its an untrusted transaction
  - This shifts authenticity function up from the IP address layer to the session/application layer
  - This devalues the incremental benefit from a secure routing system, and reduces the motivation to deploy this technology

# Why deploy BGPSEC?

- High cost, fragile framework
- The consequences of operational lapses tend towards enforced disconnection – often a far more damaging outcome than the problem that these measures they they are attempting to detect
- The network operators incur a high cost in deployment. Where is the incremental benefit? Who accrues value from this benefit?
- Partial deployments are very challenging for AS Path protection, and validating origination without propagation protection does not mitigate the security risks of malfeasance
- Is TLS doing an adequate job for the purposes we need to use it for?

# Learning from Successes

TCP

- Are we talking about TCP per se or about the use of a sliding window protocol to implement reliable packet flows in a datagram substrate?
- TCP's model of ACK-based sender clocking with AIMD-based sender governance is not ideal in any individual scenario, but is adequately good in most scenarios we've come across
- TCP's use of sender-based controls allows servers to innovate without forcing change on clients

# Learning from Successes

- BGP - The Border Gateway Protocol
  - This is a venerable protocol and continues to operate efficiently in spite of considerable scaling demands
  - What can we learn from BGP:
    - Don’t try to solve everything all at once – work with a limited set of objectives
    - Avoid flag days and forced change - use negotiation to add capabilities

# Lessons for Protocol Utility

- Design for the general case, not a specific scenario
- But focus on a single function
- Keep it simple
- Every needless exposure of data can and will be used against the user!
- Avoid inter-dependencies on third parties wherever possible
- Backward compatibility and tolerance for piecemeal deployment are essential
- Adopters need to be able to realise benefits of adoption from the outset – if benefits are realizable only when everyone deploys then adoption will stall

# Transitional Considerations

- Backward Compatibility?
- Piecemeal uncoordinated adoption?
- Early adopter advantages?
- Late adopter penalties?
- Risk reduction?
- Alignment of cost and benefit?
- Reduce dependencies?

# Lessons?

IPv6, DNSSEC, BGPSEC are all technology adoption failures in economic terms

- Economic considerations are ultimately more important than just “good” technology
- In a deregulated space, technology adoption is a function of markets, not an outcome of regulatory fiat or imposed orchestration
- Technology adoption is not based on relative merit – adoption is based on cost efficiencies, alignment of cost and benefit, and market dynamics
- The larger the system the greater the resistance to change - incumbent technologies have a strong advantage
- Displacement occurs only when the relative cost advantages are massive (10x, 100x, …)