Threats to Internet Security/Survivability

Stefan Savage
University of California, San Diego

For the non-networking/non-security folks

- The cybersecurity game
  - What input, if provided to system X, will put it in an unacceptable state?
    - Does this state satisfy goals of any adversary?
  - Few constructive theories/laws; software is arbitrarily brittle
  - "Programming the Weird Machine" – the details are where the action is
- Some networking & network security basics
  - We identify what we want to communicate with using **names** (e.g., northwestern.edu)
    - There is a distributed service (DNS) that makes these names to IP addresses
  - We deliver data using these IP **addresses** (e.g., 129.105.136.48)
    - There is a distributed service (BGP) that tells routers how to get each packet closer to its destination (even as network topology is changing)
  - Both of these distributed services rely on the correct operation of **third parties**
  - Both are subject to corruption via **impersonation**

Historic context

- Core Internet protocols **designed** for cooperative environment
  - Peers assumed to implement protocol as designed
    - IP, TCP, DHCP, ARP, SMTP, most IEEE MAC and switching/bridging protocols
    - You can lie to all of them about almost anything
  - Security left almost entirely to the future or the application
- Historic Internet protocols **implemented** in a similarly trusting world
  - RFC760 – "an implementation should be conservative in its sending behavior, and liberal in its receiving behavior."
  - Original DARPA/NSFnet routing had single backbone, BGP emerges mainly to support policy/business objectives (NOT security) -- assume all ISPs are doing the right thing
- Security emerges as a real concern in the 90s
  - In response to business needs
    - Application layer encryption/integrity – SSL/TLS (so people will do e-commerce)
  - In response to attacks
    - DDoS attacks of 99, open relay issues with SMTP, issues with source address spoofing and worms/ddos; DNS hijacking
    - RFC 1543 creates mandate for "security considerations" in protocols in 1993; lackluster effort for years; ekr writes RFC 2552 on how to write such sections in 2003
      - All well after key protocols have been standardized and deployed

Where is most Internet security effort over the last 20 years?

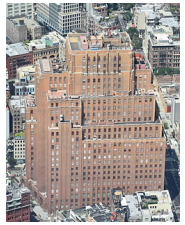
- TLS and the CA-based PKI infrastructure
  - Significantly driven by browser vendors
- Fixes to individual protocols where big vulnerabilities are being exploited
  - E.g., Bailiwick checking for DNS cache poisoning and then QueryID hacks in response to Kaminsky
  - SYN cookies for SYN flood DDoS, etc.
- Largely unsuccessful efforts to secure DNS and routing
  - DNSSEC, SBGP
  - RPKI and MANRS
- Lots of academic papers on lots of other things

Issues

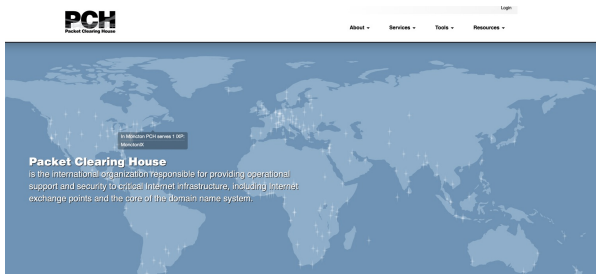
- Centralization magnifies impacts
- Decentralization is complicated, untrustworthy and hard to audit
  - Dependencies are complex and unknown
  - Trust doesn't scale
  - Integrity failures can be invisible
- Key protocol/service deployments are not well-tested against threats
- DDoS will always be with us

Centralization

- Economic forces encourage centralization
  - Amplifies impact of failure or attack
- Physical network infrastructure
  - Fiber, switching, interchange, etc
  - Cell backhaul and towers
  - Control over same (e.g., only really 3 cell carriers in US); iconectiv for number portability



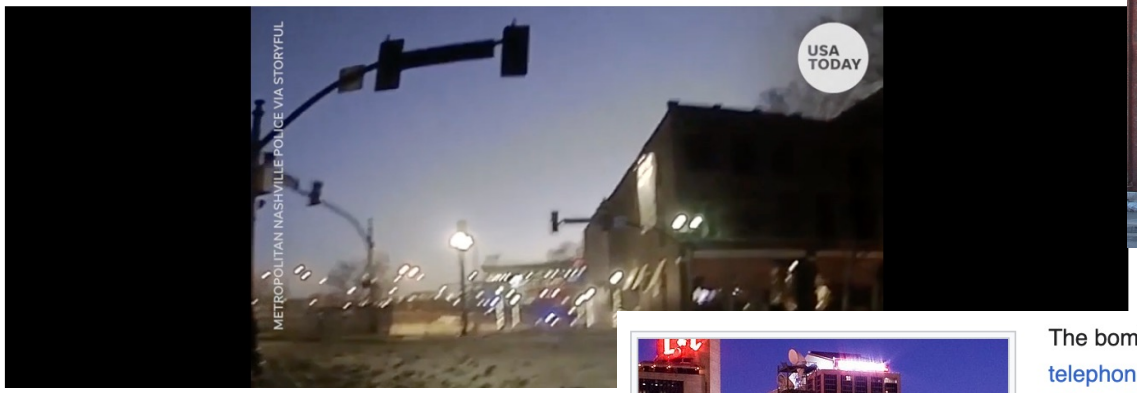

- Cloud services
  - Six companies deliver the majority of Web resources in Alex 1M [Doan et al, TOIT '22]. 1 of 3 scripts
    - Huge *internal* networks that can frequently skip traditional transit providers
  - Top 3 DNS, CA, and CDNs cover between 50-70% of top 100k sites
  - Handful of operators run all the big gTLD registries
  - Public resolvers (e.g., 1.1.1.1, 8.8.8.8) centralizing DNS resolution
  - Microsoft and Google handle email for ~30-40% of all domains

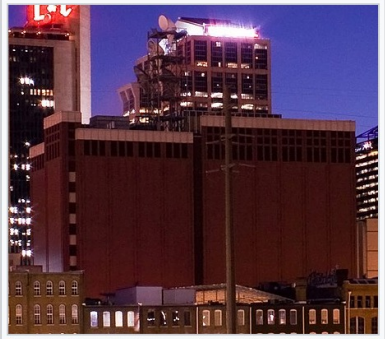


Nashville bombing froze wireless communications, exposed 'Achilles' heel' in regional network

Yihyun Jeong and Natalie Allison Nashville Tennessean

Published 10:07 a.m. ET Dec. 29, 2020 | **Updated 11:23 a.m. ET Dec. 29, 2020**





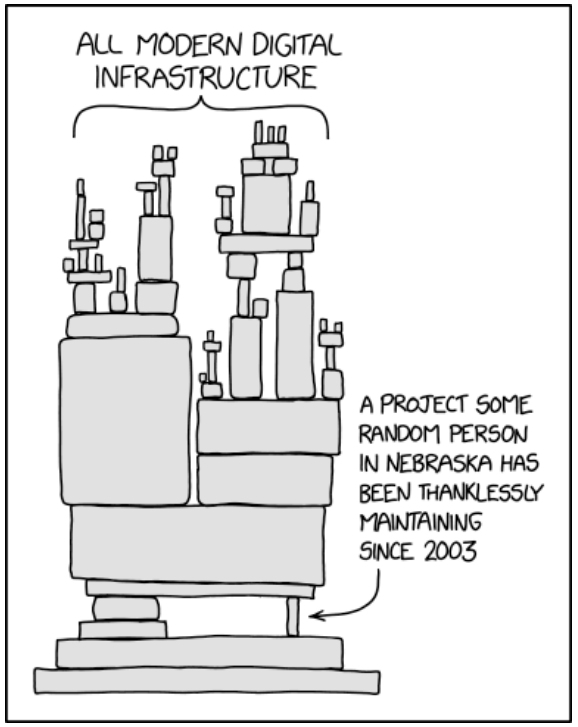
The AT&T facility in front of which the bombing occurred, pictured in 2009

The bombing caused structural and infrastructure damage to a nearby AT&T service facility, which contained a telephone exchange with network equipment in it, resulting in AT&T service outages across the U.S., primarily in Middle Tennessee.[26] Although the facility's backup generators were rendered nonfunctional because of fire and water damage, communication services initially remained uninterrupted while the facility was able to run on battery power.[27] However, outages were reported hours after the explosion, with significant service disruptions in the area by around noon.[26] Cellular, wireline telephone, internet, and U-verse television service were affected, as were multiple local 9-1-1 and non-emergency phone networks in the region, along with Nashville's COVID-19 community hotline and some hospital systems.[6][26][28] T-Mobile also reported interruptions to its service.[29] The Memphis Air Route Traffic Control Center experienced communication issues, leading the Federal Aviation Administration (FAA) to ground flights from Nashville International Airport for about an hour.[30][31]

Outages continued to affect communication services, including Internet, phone, and 9-1-1 services, for days after the bombing.[32][33] Some stores reported switching to a cash-only policy because credit card systems were out of service, and issues with ATMs were reported.[34][35] AT&T mentioned deploying two mobile cell sites downtown by the next morning, with additional ones deployed throughout Nashville by evening, but it gave no specific timeline in regard to a full restoration of service, adding that a fire that reignited during the night led to an evacuation of the building.[33][34] Officials later said a full service restoration could take days.[21]

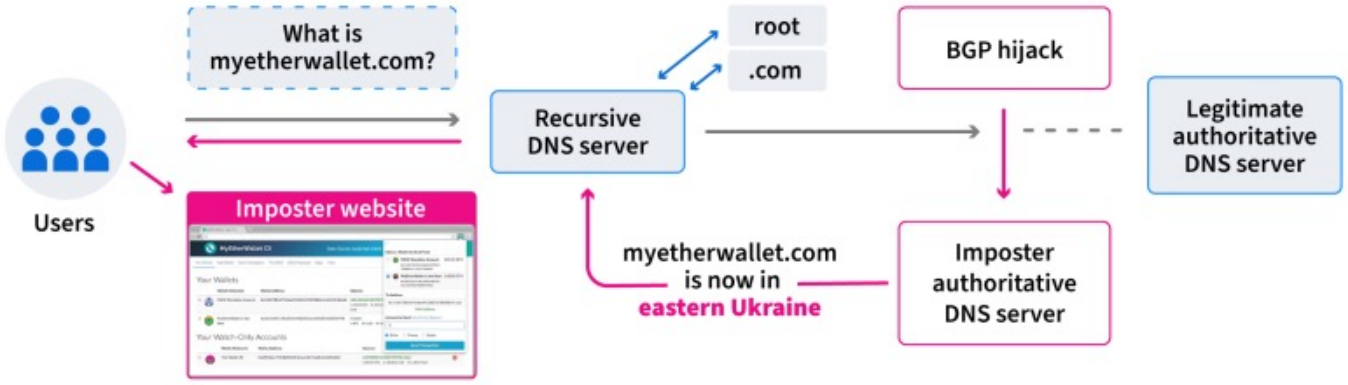
Dependencies are complex and unknown

- Systems increasingly inter-dependent
  - Cloud compute/hosting
  - Web infrastructure
  - Internal systems-as-a-service
  - Key services (e.g., time)
- No straightforward way to establish dependency graph
  - Where do my two ISPs have separate physical infrastructure?
  - If AWS goes down, could that impact my network provisioning system?
  - There are tons of "post mortems" full of such surprises
  - i.e., its not in the architecture
- No real composition architecture for cloud services
  - Lots of vulnerability at the interface between
- Lack of resilience is invisible – until failure



Trust doesn't scale well

- Inevitably, for integrity, we want to establish statements that relate some claim to some real-world property (e.g., when I go to amazon.com its Jeff Bezos' shop)
- Our solution is to pass the buck to someone else and trust them
  - CAs (200+) – some limited due diligence or domain control evidence – claim signed cryptographically
  - Registrars (2500+)– zero due diligence, claim controlled by limited access to EPP for given registry
  - IRRs (5+15)– some limited due diligence (email in whois!), some cryptographic signing (RPKI), but pretty open NRTM
- Consequence – everything can fall apart if one trusted entity gets compromised

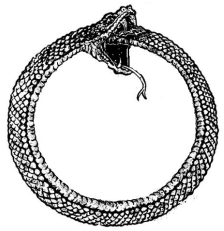


Integrity failures can be invisible

- Decentralized protocols (e.g., DNS, BGP, CAs, etc) have great scaling properties but are challenging to audit – no single state

- What happens if northwestern.edu was poisoned in a DNS resolver cache for Comcast in Atlanta?
  - How would you know?
- What happens if 129.105.0.0/16 (northwestner's network) was hijacked for but only for CENIC/CalREN?
  - How would you know?
- What if these happened for only 10 minutes?

The authentication ouroboros



- We know DNS and BGP are vulnerable, so we rely on end-to-end integrity via TLS
  - TLS validates that the other party has a valid certificate, signed by a CA, for the domain name
- LetsEncrypt and other CAs use domain validation to provide cheap due diligence for awarding new certificates
  - If you can hijack IP space of A record, or for NS server you can get valid CA
  - Or (easier) you hijack the NS record directly (by compromising registrar account or registry)
  - Transforms DNS/BGP capability into valid cert, undermining value of TLS
- This happens (see Akiwate et al, IMC 22) but its very hard to **tell** that it happened
- Current Internet architecture not designed for auditability
  (But Certificate Transparency is a step in the right direction)

Key protocol deployments are not well-tested against threats

- Sometimes its because they are key production protocols
  - BGP – how well would current Internet weather a handful of ASs flapping 100k routes?
    - what would happen if Google injected routes for all of AT&T and Verizon's customers? (what about non-customer routes?)

- Sometimes its because they are proprietary implementations
  - E.g., protocols use to replicate state inside Akamai, CloudFlare, Amazon, etc

- Or they are somewhat "invisible" and with limited access
  - EPP is a great example; one of the invisible "back-end" protocols that run the show
  - SS7 underneath everything
  - An array of provisioning protocols in cellular, HFC, and Cable networks

DDoS

- The solutions aren't much different than they were 20 years ago
  - Divert and to expensive box and clean if there is a clear content pattern
  - Divert to CDN and spread load (but someone needs to pay)

- Still no cost-effective way to manage large-scale wanted vs unwanted traffic outside your own network infrastructure

Some ultimate issues

- Centralization is cheap and useful; but magnifies rare failure
- Decentralization supports innovation and expansion, but creates transitive trust relations and hence easier to attack
- We have no good theory about where to use one vs the other

- We have very limited visibility which hamstrings both design for resilience and detection/triage of problems