The Tragedy of Trust: Network Protocol (in)Security

Nicholas Weaver
International Computer Science Institute

Who Am I?

- I am a researcher at the International Computer Science Institute in Berkeley
  - ICSI is a nonprofit research lab affiliated with the university
- My primary area of research is network security:
  - Worms, malcode, intrusion detection, etc etc
- I'm also notoriously paranoid and with a very devious mind:
  - "My Evil Twin" is my threat model: an adversary who is as capable, creative, and devious as possible.

This Lecture:

- The fundamental problem on our network: Most protocols date back to a nonmalicious era
  - What can be done as a man-in-the-middle?
- The Border Gateway Protocol (BGP)
  - Internet Routing 101
  - BGP Blackhole attacks
  - BGP Man-in-the-Middle attacks
- The Domain Name Services protocol (DNS)
  - DNS 101
  - DNS Cache poisoning
- Key discovery:
  - The Secure Shell protocol (SSH)
  - HTTPs (Public Key Infrastructure)

A Brief History of the Internet…

- TCP/IP: 1973-1978
  - How packets traverse over networks
- Ethernet: 1973-1976
  - The physical media for attaching computers
- Domain Name Service (DNS): 1983
  - How to find a computer's address
- Border Gateway Protocol (BGP): 1989
  - How to discover packet routes
- Address Resolution Protocol (ARP): 1982
  - How to find other hosts on the local network
- Dynamic Host Configuration Protocol (DHCP): 1993
  - How to find your own address on the local network
- All these fundamentals were designed for ***non-malicious*** networks

Common Goal of Most Attacks

- Denial of Service:
  - Prevent someone from performing an operation
- Eavesdropper:
  - See all traffic but not modify traffic
- Man-in-the-middle:
  - See and modify all traffic
- And then convert that into a ***benefit to the attacker***
  - Attackers don't act without reason, there must be at least some motive

The Interdomain Routing Problem: BGP

- The Internet is composed of numerous connected Autonomous Systems (ASs) which are independent networks connected together
  - If the destination of a packet is within the current AS: just forward it through the internal destination
  - But if the destination is external, how do we know where to send it?
- The Border Gateway Protocol (BGP): a method for an AS to notify everyone else what networks belong to this AS, and to know how to direct any traffic towards the correct destination
- Note: routing is based on ***netblocks***:
  - 192.169.0.0/24: All addresses between 192.169.0.0 and 192.169.0.255
  - 192.169.4.0/22: All addresses between 192.169.4.0 and 192.169.7.255

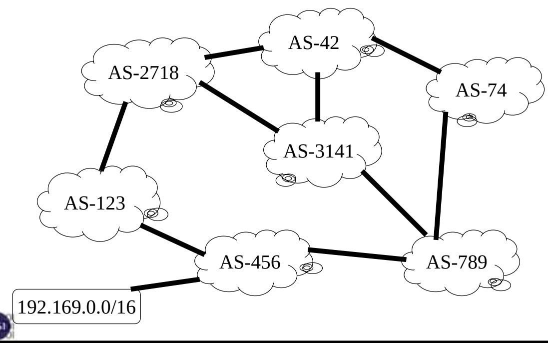
How BGP works:

Network Protocol (in)Security Nicholas Weaver

- If an AS is responsible for a netblock, it "advertises" this netblock to the neighboring ASs
  - This says "I can accept all traffic for this netblock"
- If an AS is willing to provide ***transit*** for another AS, it will advertise the netblock as
  - "I can accept all traffic for this netblock, and it will pass through me and this path of ASs to the final destination"
- If an AS sees multiple advertisements
  - Choose the most specific:
    - 192.169.0.0/24 will take precident over 192.169.0.0/23 for packets going to 192.169.0.23
  - Choose the shortest path
    - No loops

BGP 101:
Animated: See Whiteboard

Network Protocol (in)Security Nicholas Weaver



BGP Blackhole Attack

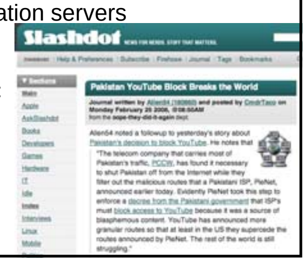
Network Protocol (in)Security Nicholas Weaver

- Step one, get a peering arrangement with ***somebody***
  - Become an AS, hard but not THAT hard
- Now simply advertise a more specific route: If your victim is 192.169.2.34 in a /16 netblock, advertise a route for 192.169.2.0/24, and your route takes precedent
  - Even in the case of a tie, you can still capture/deny for all ASs closer to you than your victim, since BGP selects the shortest AS path

DNS Blackhole Attack
in the Wild

Network Protocol (in)Security Nicholas Weaver

- This actually happens, often by accident: About a year ago, YouTube was blocked because a pakistani ISP advertised the routes for YouTube's coordination servers
  - Resolution involves human mediated detection and response: Find the upstream point of the bad AS and get them to stop accepting the bad route



Why Does this Work?
Abuse of Trust

Network Protocol (in)Security Nicholas Weaver

- All ASs have to trust their neighbors, which trust their neighbors, which trust their neighbors…
  - So all it takes is ***one*** AS which mistakenly trusts a malicious AS that it peers with
- Trust in BGP is ***transitive*** and ***global***
  - ***Any*** system with global transitive trust is subject to such abuse

But Blackhole is Not
That Useful

Network Protocol (in)Security Nicholas Weaver

- Its only a Denial of Service:
  - Allows you to knock someone off the net, not monitor their traffic
- It doesn't last that long
  - People ***notice*** their traffic is dropped
  - RouteViews or similar tools (show BGP behavior) can find the offender(s)
  - Offender's upstream contacted to drop the offenders
- Thus more likely to happen by screwup rather than malice

Turning Blackhole Into a (one sided) Man In The Middle

- The Polokov attack:
  - Performed ***live*** at DEFCON 2008:
    ALL traffic returned to DEFCON passed through Texas…
- Simple addition to the Blackhole Attack:
  - Have TWO connections to the Internet:
    One with a full peering connection (the attack link)
    One that doesn't filter packets by IP address (the return link)
- Through the return link:
  - Perform a traceroute to your victim's network: Compute the AS path for this route (the ***return AS path***)
- Through the attack link:
  - Advertise your victim's network (as a blackhole), but ***prepend the return AS path***
    - Now all ***but*** the return AS path will direct traffic to you
- And modify the packets…
  - When you receive a packet to the victim, ***increment*** the time-to-live field and forward it through the return link

General Countermeasure: Monitoring

- Multiple services offer pictures of the current BGP feeds
  - ***Routeviews*** service
- Use your link and a backup link to ***monitor*** these remote BGP feeds
  - If ever ***your*** networks are not showing the proper route, ***alert*** someone responsible
    - The network operations crowd is a very small community, everybody knows who to call when there is such a problem
- Limitation: ***not*** instantaneous
  - May take a few hours to resolve problems

General Countermeasure: Ownership/Authentication

- A lot of work has been put into place in trying to keep track of who owns what…
  - Perhaps with cryptographic authentication
- Problem: BGP thrives on flexibility
  - Multihoming: Advertise routes through 2+ ASs to provide better performance/reliability/lower-cost.
  - No lockin: Easy to shift to different transit providers
- Problem: Legacy
  - Routers are not that flexible: adding crypto overhead is a worry

The Domain Name Service (DNS) Protocol

- The Internet operates in IP addresses…
  - But people think in names
- DNS turns names into addresses
  - www.foo.com is 10.0.32.14
- System is ***heirarchical trust***:
  - Top level (.) ***roots***
  - Top Level domains (TLDs), eg, .com, .org, .gov
  - Second level domains, eg, foo.com, bar.gov
    - Can nest arbitrarily
  - For everything within foo.com, you need to trust foo.com's nameservers, .com's nameservers, and the root nameservers

DNS Illustrated

- See Whiteboard:
  Stub Resolver: Your System
  Recursive Resolver: The ISP's central DNS server
  Authoritative Servers: Systems which own the domains
- Responses include 4 groups of records:
  - QUESTION: what was the question
  - ANSWER: what are the answers
  - AUTHORITY: what are the authoritative servers
  - ADDITIONAL: any additional mappings
    - IP addresses of the authoritative servers
    - Other useful addresses
  - Authority/additional records are commonly called ***glue records***

Authoritative/Additional Data: Old-School Cache Poisoning

- DNS resolvers don't ***just*** cache the response: they also ***opportunistically*** cache the glue records
  - Otherwise, a subsequent fetch would requiring going all the way back to the root
- What happens if the authoritative or additional fields are ***incorrect?***
  - EG, if the response for www.foo.com, contains an additional record saying www.bar.com is 127.0.0.1?
  - A recursive resolver would ***accept*** and ***cache*** the response, and now any further request for www.bar.com would return the wrong value

Poisoning: Bailywick Checking

- Often cache poisoning occurred by ***accident***
  - Eg, the authoritative server for foo.com was misconfigured
- Solution was ***bailywick checking:***
  - ***ONLY*** cache authoritative or additional data if ***within*** the authority of the server:
    EG, for .com, will accept and cache any returned value that ends in .com
    for foo.com, will ***only*** accept and cache returned values that end in .foo.com

The Small Transaction ID: Old-School Blind Injection

- DNS uses UDP, not TCP
  - Protocol is ***connectionless***
  - Only check is that the response is consistent***:***
    - Comes from the correct server, with the correct ports, and the correct 16 bit transaction ID
- For most server, the only thing which varies is the transaction ID
- Attacker tricks the ISP's resolver into looking up an address (eg, www.foo.com)
  - At the same time, sends a bunch of responses of the form: www.foo.com is my.evil.address
  - If the transaction ID matches, the resolver accepts the attacker's response
- Now attacker can be a full ***man-in-the-middle***: all traffic is redirected through the attacker's server

Long Known but "No Worry"

- Attack could only be attempted once per TTL
  - Until the TTL on the legitimate entry expired, the attacker couldn't try again
- Most ***important*** names have long TTLs
  - The names and addresses of the TLD (Top Level Domain) servers, eg, .com, .org, .gov, etc…
- But even so, odds are not comfortable:
  - An attacker could easily send 1000 packets in an attempt: Odds of success are 1-(1-2^-16)^1000: or about a 1.5% chance of success

The *Kaminski* variant: Achieve *Race-Until-Win*

- Instead of trying to poison www.foo.com, try to poison 1.foo.com
  - ***But*** have the response include an additional record saying www.foo.com is attackers.evil.server
- If success, great!
  - The response is ***in bailywick***, so it is accepted
- If failed, try to poison 2.foo.com….
  - Just keep trying different names until one is successful!
- But you can do ***even better***:
  - Try to poison 1.com, 2.com, 3.com…
  - In the response, say the ***authority*** for .com is the attacker's NS server
  - Now ***all*** subsequent DNS lookups are controlled by the attacker!

Defense #1: Increased *entropy*

- Instead of always using the same UDP source port, select a random source port:
  - Attacker needs to guess ***both*** the transaction ID and the source port used:
    This significantly reduces the odds of succes (1 in 2^30 instead of 1 in 2^16 per packet…)
- 0x20 randomization:
  - DNS is case insensitive: www.foo.com is the same as wWw.FOo.cOM
  - But almost all authorities preserve case (lazy programmer just bitwise-copy the question)
  - Thus randomly apply a capitalization

Defense #2: Detection and Response

- Easy to detect: Look for responses with wrong transaction IDs
  - Need to increase entropy first, because the odds of missing an attack are too high without increased entropy
- A ***possible*** response that might actually work:
  - Generate ***two*** identical requests with different entropy: Accept them only if the two responses match
    - Attacker would have to win two simultaneous races: effectively doubling the entropy

Defense #3: Glue Policy

- Entropy defenses increase the attacker work in ***packets***, a different glue policy increases the attacker work in ***time***:
- One such policy:
  - Accept ***ALL*** glue for the purposes of resolving the current transaction
    - Necessary to ***resolve*** a name
  - ***ONLY*** cache the direct response to the question
    - ***Prevents all*** race-until-win attacks on a given name, as queries ***will never*** be generated as long as there is a valid cache entry
  - ***Independently fetch*** any glue records not currently in the cache
    - ***Future queries*** will have the same advantage of a full cache
- Results in increased load but no other effects
  - Except for a few servers

Defense #4: DNSSEC

- DNSSEC is a protocol for cryptographically signing DNS records
  - A ***data integrity*** protocol
- Operates on the same tree of trust as DNS:
  - ***Roots*** sign a domain's key which can sign a sub-domain's key…
  - Unfortunately, there is a big political question: who will sign the root?
    - Thus only

The Bigger Problem of DNSSEC

- DNSSEC is designed to target ***in path*** adversaries the other defenses prevent only ***out of path*** adversaries
  - But such attackers really target the final protocol:
- If the protocol trusts DNS, it trust the network
  - Thus securing DNS offers no benefit
- If the protocol doesn't trust the network, it never trusted DNS
  - Thus securing DNS offers no benefit
- The real benefit: a ***lower cost*** Public Key Infrastructure
  - Rather than paying for a public key per server-name, you pay once per domain and can generate your own subkeys

So what *should* a network protocol assume…

- Trust ***as little as possible***:
  - ***Assume*** the network is an adversary
- Be explicit in what you ***do*** trust
- Use ***public key cryptography*** to ensure ***integrity and confidentiality***
  - Public key allows two systems
- But you somehow need to learn the remote host's public key…
  - This is the key foundation of trust in a real network protocol

Key Learning: ssh

- Key idea on ssh: you only need to trust history
  - The first time you contact a remote system, you accept the public key
    - A leap of faith
  - Subsequent connections ensure that the public key doesn't change
- Thus you can only be man-in-the-middled on the first time you connect to a remote host
  - As long as the first connection was safe, its OK
  - And if paranoid, you can use an out-of-band way of confirming the fingerprint

Key Learning: CAs and PKIs

- Public Key Certificates:
  - A public key and associated data (eg, what host, what individual) cryptographically signed by ***somebody***
- Certificate Authorities:
  - An authority which signs a bunch of certificates
- Public Key Infrastructure:
  - A chain of certificate authorities
- Creates a ***tree*** of trust from one or more roots
  - Concept is used for ***ssl (https)***: your web browser has a list of certificate authorities