Architecting a Secure Internet

Saikat Guha, Paul Francis
Cornell University

ABSTRACT

The Internet is not secure due to its design goals being at odds with the principle of least privilege. The Internet strives to allow any host to communicate with any other host, while the principle of least privilege advocates limiting host connectivity to the smallest set necessary for performing a task. Our goal is to secure the Internet by largely turning off connectivity in the Internet, and then using explicit signaling to selectively enable only those connections that are deemed necessary for performing a task.

Intrusions and worms routinely exploit application vulnerabilities, DDoS attacks exhaust a target's network bandwidth, and phishing attacks and spam trick users on a daily basis. This is in no small part due to the design goals of the Internet being at odds with the principle of least privilege. Firewalls embody this principle in spirit by restricting access between hosts. Being an ad hoc bolt-on to the original Internet architecture, however, firewalls have limited knowledge (just IP addresses and ports) on which to base their decision to block a packet. In the client/public-server Internet where a client can typically access any public service, this limited knowledge is sufficient for restricting connectivity but it is not the case in the client/client Internet popularized by VoIP, P2P filesharing, VPNs etc. As client IP addresses change (DHCP, VPN, NAT) and applications use dynamic source and destination ports, firewalls are left guessing as to the intent of a connection and whether or not to block it.

Our goal is to secure the Internet by allowing firewalls to enforce the principle of least privilege in all cases. This involves incorporating multiple layers of firewalls into the Internet architecture, which corresponds to the role multiple parties (end-user, IT Department, ISP) play in allowing a connection. Our approach leverages the existing IP routing core unmodified but constrains host connectivity by configuring firewalls to disallow all connections by default. To establish a client/client connection, an application must explicitly request permission from all the firewalls between it and the remote end-point by declaring the intent of the connection over a signaling channel (like SIP, the Session Initiation Protocol used for signaling to establish VoIP calls). The request is rich enough to allow firewall administrators to target individual users and applications, and negotiate security parameters for the resulting connection. This creates the need for richer naming and connection semantics that goes beyond the 5-tuple (protocol, IP addresses and ports) used today to include user credentials of both end-points, application integrity, and even ACLs or labels for data to be transmitted over the network. The framework is robust enough to allow the firewall administrators to require hardware attestation at the end-host and negotiate the use of encryption for the data channel based on the level of security required. Finally, our approach performs late-binding between the signaling channel and the data routing path whereby applications pick transport addresses and firewalls are configured to accept data packets for those addresses (and enforce the policy negotiated) after the connection request has been granted. Effectively, our approach largely turns off connectivity in the Internet, and then selectively enables only those connections that are deemed necessary for performing a task.

From the users' and administrators' perspective, our architecture secures the Internet by giving users control of connectivity to and from their applications, and network administrators control of what users and applications can use the network infrastructure and in what manner. Using the primitives provided, a network administrator can restrict user `alice@cs.cornell.edu` using mail-reader `microsoft.outlook` to be able to connect only to the department mail-server `microsoft.exchange` being run by `admin@cs.cornell.edu`, and nothing else. Meanwhile, user Alice can restrict any application on her computer that read data labeled `alice.confidential` to be able to contact only those other applications that are also being run by Alice herself and require that the connection be encrypted. While the first rule prevents a zombie mail-reader from abusing the network by spewing spam or worms, the second rule prevents a trojan from stealing Alice's passwords.

While the idea of securely architecting the Internet by having firewalls turn it off is interesting and heretical to some extent, the key technical challenge lies in defining new abstractions and services that allow legitimate applications to use the network. To that end, there are many open questions. First, what abstractions and guarantees should the network provide to the OS? What should the OS provide to the network? For example, the network (with help from the remote OS) could attest to the local application that a particular TCP connection terminates at a particular application being run by a particular user, is encrypted and secure, and has guaranteed QoS at the gateway. Second, how rich do the naming and connection semantics need to be? Strict policy on IP addresses and ports can prevent today's worms in client-server applications but are ineffective for securing P2P applications where both end-points use dynamic IPs and ports. Using application names and restricting connectivity to a small world of trusted users does slow worm-propagation, but true containment requires hardware attestation. Third, does transferring the security problems from the Internet to the signaling domain, as we suggest, really solve them? The signaling domain is much simpler and provides a framework for negotiating trust before a connection is established, but it does require a trusted firewall. We do not have all the answers yet, but we have opened up an interesting design space for the next-generation secure-Internet architecture.

SOSP'05, October 23–26, 2005, Brighton, United Kingdom.

Problem: Internet Security

- The Internet is not secure
  - Worms, Viruses, Intrusions, Phishing, Spam, DDoS, ...
- Violates **Principle of Least Privilege**
  - Any host can connect to another
  - Even if it doesn't *need* to

Existing Solutions: Insufficient

Firewalls ...

- Ad hoc
- Operate in the dark (IP:port meaningless nowadays)
- No security for public client-to-client Internet

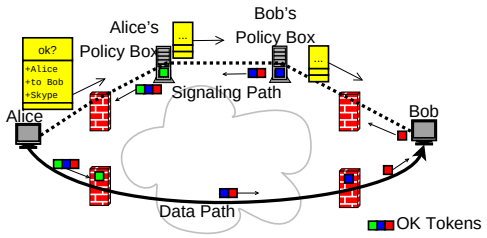
Ideal Solution ...

- Enforces principle of least privilege
- Incrementally deployable

Solution: Signaling

- Clients *cannot* talk to another client by default
- Enforced by firewalls at endhost, corporate gateway, ISP ...
- Clients signal *intent to communicate*
  - User and application credentials
  - Security parameters, etc.
- Firewalls allow connection that don't violate local policy.

Solution: Signaling



"Replace democratic any-to-any Internet with royal palace of mediated interactions"

Related Work

- Is the Internet Going NUTSS? [Francis, IntComp'03Nov]
- SIP based approach to connectivity [Guha, FDNA'04]
- DoS-resistant Architecture [Handley, FDNA'04]
- Delegation-Oriented Architecture [Walfish, OSDI'04]
- Off by Default [Ballani, HotNets'05]

Our approach: Explicit signaling, no heuristics, incrementally deployable without new infrastructure

Work In Progress

- Richness of naming, intent-to-connect
- Help from the endhost OS
  - Hardware attestation
- Usefulness
  - Trusted network
  - *Global* Asbestos [Efstathopoulos, SOSP'05]

`http://nutss.net/`