User Consent and Security as a Public Good

Patrick McManus, Firefox Networking Lead
Richard Barnes, Firefox Security Lead

The “workshop bio” for MaRNEW observes that the increasing use of encryption in the Internet is “a good thing for consumer and business privacy and security”, and notes that “business-related topics” can be included under the heading of network management.

Given this framing, it may be tempting to look at security issues through the lens of small-scale, provider-customer business concerns. After all, the user has established a business relationship with his access ISP, and that relationship entails certain obligations on both parties. The ISP will provide Internet service, and the customer generally agrees to an acceptable use policy that prohibits illegal or abusive use of the network.. There is already a negotiation between the customer and the ISP happening here. It might thus be tempting, but a bad idea, to use this customer/ISP relationship as a place to negotiate security policies for a user’s applications too.

In this paper, we will discuss a few ways in which this myopic focus misses important details of how the Internet works today. Security policies applied at the application layer do not just have an effect locally. They apply to the application’s traffic all across the Internet, and they can have ramifications for every other host on the Internet. So it’s critical to regard security as a common, Internet-wide value. In this way, security and user choice go hand-in-hand. Security technologies don’t preclude intervention by the network, but they do require that these interventions be explicit and authorized by the communicating parties.

Security is a common good[1]

In order to provide value to the various parties participating in it, the Internet needs to provide a sufficient level of security to prevent bad actors from interfering in communications. The security of the Internet emerges from myriad local decisions, by end hosts as well as network operators. These local decisions, however, can have global ramifications across the network — and these aggregate security properties of the entire network are in turn critical to each application

Consider IP address spoofing as an example. Networks that allow IP address spoofing are a major problem for the security and stability of the DNS because botnets in such networks can use the DNS to amplify[2] a DDoS attack. The IETF published guidance[3] on preventing spoofing

[1] https://www.w3.org/2014/strint/papers/60.pdf
[2] https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/
[3] https://tools.ietf.org/html/bcp38

back in 2000,but significant portions of the Internet[4] still lack protection against IP address spoofing. Local decisions by network operators not to prevent spoofing lead to DDoS risks for everyone else on the network.

Recent attacks have demonstrated similar principles at work at the application layer. In the "Great Cannon" attack[5], network operators were able to cause a DDoS by injecting JavaScript code into unencrypted web pages (i.e., those not using HTTPS). Similar techniques[6] were used by Hacking Team to inject malware from inside an ISP network, allowing them to infect users and subvert the TOR network. Local decisions by web sites not to use HTTPS lead to increased DDoS risks and user vulnerability at many points across the network.

Another noteworthy feature of the above attacks on the web is that they were accomplished from deep inside the network, not the edge. The Great Cannon attack was able to achieve scale by operating at the boundary of the Chinese Internet. Hacking Team's injection utilities could run just as well in a transit provider's network as an access network. Even users of highly trustworthy access ISPs would be vulnerable to these attacks as soon as they communicate with the broader Internet.

In considering how to make trade-offs between security and optimization, it is thus important to consider the implications of these decisions beyond the first hop. Non-secure connections are vulnerable all the way across the network, from the first hop, across transit providers, all the way to the remote host. And a weakness at any of those points can be turned into threats against the whole network.

A gap in security anywhere is a threat to hosts everywhere. In order to provide an overall level of security for the Internet, the local decisions made when building networks and applications need to be secure by default.

Security makes user consent fundamental

All of this security does not mean that there is no way for network operators to affect the applications that are carried over their network. Network security just interposes a requirement that the negotiation between the user and the network operator be explicit and secure.

Security technologies exist to draw lines — to verify the identities of the parties to a communication, to ensure that the communications are confidential to those parties, and to

---

[4] http://spoofer.caida.org/summary.php
[5] https://citizenlab.org/2015/04/chinas-great-cannon/
[6] http://motherboard.vice.com/read/hacking-teams-spyware-targeted-porn-sites-visitors and https://www.documentcloud.org/documents/2166407-hacking-team-project-x.html#document/p11

prevent anyone besides those parties from tampering with the communication. That is, security technologies lock out unauthorized parties.

Security technologies do not define where those lines should be drawn, which parties are authorized. A secure connection doesn't mean that everyone is locked out, just that only the right ones are in. If the parties to a connection authorize further parties to have access, then it's largely just a matter of bookkeeping to enable that party to have access.

"Cooperative management" is still possible in a world where most communications are encrypted; it just needs to be authorized by one of the "first parties" to the communication. In designing such "cooperative management" technologies, however, there are a few important design constraints to note.

Most obviously, the agreement itself needs to be conducted securely. Any time an additional party is added to a secure communication, the mechanism must ensure that the new party is authenticated. Any mechanism that would allow an unauthenticated party to have access to a secure communication is missing an important security feature, and it will be abused by bad actors.

If security requires user consent as a prerequisite for cooperative management, then there must be mechanisms for acquiring and expressing this consent. User consent can be manifested either by an action by a real, human user, or by an action taken on the user's behalf by a user agent. So the trade-offs involved in such a decision need to either be obvious enough that there's a clear default decision for the user agent to make without involving the user, or they need to be expressed at a high-enough level that a non-technical user can make an informed decision. In addition, it must be possible to make these decisions at a fine-grained level (e.g., applying to some applications and not others), and the consent must be revocable.

In designing these trade-offs, it will be useful to keep in mind a doctrine of least privilege. The more access a network function requires to secure content, the more difficult it will be to convince users and user agents that they should allow that network function access. Reducing these requirements will reduce barriers to cooperation. For example, the IETF PERC working group is working to create conferencing systems where the centralized server can provide switch media packets without having access to encrypted media. There have been some discussion of HTTP caches that can reduce duplication of content without having access to the content, and advanced AQM mechanisms[7] that allow bottleneck management of data without access to its contents

[7] https://tools.ietf.org/html/draft-ietf-aqm-fq-codel-01

Conclusions

The orderly operation of the Internet as a public resource requires applications to be able both to communicate reliably with parties of their choice, and to protect their communications from other, unauthorized parties. The best techniques we have to achieve these goals is thoughtfully-applied cryptography, integrated into Internet applications.

Experience has shown that network middleboxes that seek to optimize this experience above the network routing level, without the consent of the end points, may provide brief point in time optimizations but over time introduce ossification, security vulnerabilities, and fragility into the network[8]. In the end they often end up being a net loss.

On the other hand, it is possible to create intentional arrangements that explicitly integrate network infrastructure into the application architecture. These arrangements will need to include explicit consent by the user (or the application on his behalf), but once they are available, they can allow allow both network operators and applications to benefit from network-provided optimizations when appropriate.

[8] https://tools.ietf.org/html/draft-hildebrand-middlebox-erosion-01