STRINT Workshop Position Paper

Jon Peterson

Neustar, Inc.

jon.peterson@neustar.biz

***Abstract*—While SIP is widely used as a protocol for real-time communications, it is very difficult to secure from pervasive monitoring. In fact, one could argue that SIP's susceptibility to mass surveillance was essential to its success in the marketplace. This paper shows why SIP's design left the door open for eavesdropping, and what lessons RTCWeb could learn from this.**

I. Introduction

SIP, like many application-layer protocols, offers several optional security services—though historically these have been used only sparingly. Even those mechanisms that the IETF mandates to implement have spotty deployment records and little prospect for interoperability, as operators and users rarely choose to enable these features. There are notable success stories, like Digest authentication as a means for users to identify themselves to services, or to a lesser extent TLS for securing hop-by-hop connections. But other security services, like true end-to-end confidentiality for personal communications, have remained elusive despite numerous attempts to render such systems as opportunistic as possible. The most frequently blamed obstacle historically has been key management: both enrolling endpoints in the keying systems confidentiality requires, and then deploying the infrastructure to discover keys in a timely manner.

But to many SIP operators, security is often seen as just another thing that can go wrong. As with other applications, users rarely clamor for privacy-enhancing features, and more significantly, even after high-profile security lapses became public scandals, users seem willing to tolerate services which lack substantial privacy protections. Also, were strong security in place, any false positives due to misconfiguration or other forms of human error would annoy users and potentially alienate customers. Accordingly, operators strive to transfer the burden of security to the service, through intermediaries, rather than pushing security information down to endpoints. This has two important consequences: first, that users typically have no way of knowing whether or not security services are in place; second, that operators are parties to any trust relationships that users establish.

II. SIP—Fantasy and Reality

During the design of SIP, there was significant controversy about the powers that SIP should grant to intermediaries. Drawing on personal communications architectures similar to email, SIP effectively requires an endpoint to register with an intermediary (a proxy server) in order to receive SIP calls; e.g., in order to receive calls for the "address of record" URI sip:alice@example.com, Alice's endpoint must register with the example.com SIP service. Such intermediaries implement a stateful location service for associating endpoints with an address of record when registrations are received, and then route incoming requests for that address of record to those endpoints. To perform this function, the SIP standard (RFC3261) stipulates that intermediaries may alter the incoming request in a number of respects, including:

- They may change the Request-URI from the address of record URI to a URI designating the endpoint, or any other URI the service chooses (which could for example be the address of record of a different user, in a call forwarding case)
- They may add a Via header (comparable to the "Received" header of email) which is used to route responses in the backwards direction
- They optionally add a Record-Route header which is used by endpoints to build a Route header set; Route header sets designate which intermediaries future requests in a dialog will transit

RFC3261 explicitly designated which components of a request intermediaries were permitted to change; it explicitly forbid intermediaries from altering messages bodies, for example. In practice, however, intermediaries alter messages in much broader ways than RFC3261 envisioned. For example:

- They remove any unrecognized or unauthorized SIP headers or option-tags, in order to prevent interoperability failures or the enablement of features that contradict local policy
- They change the Contact header field, in order to obscure any potential privacy sensitive information that could be leaked by the Contact URI (some devices build the Contact URI using the IP address of the endpoint), and also to prevent endpoints from sending traffic to one another directly when that contradicts local policy
- They change the Call-ID, as the Call-ID is created by some legacy devices through concatenating a string containing the endpoint's IP address, with its potential privacy sensitivities

- They remove Via headers in order to provide further "topology hiding," that is, to prevent leaking to untrusted endpoints details about prior intermediaries in order to keep these secret, either to protect business arrangements or to obscure targets from potential hackers
- They remove Record-Route or Route headers for similar reasons, and moreover to prevent endpoints from routing future requests in ways that contradict local policy
- Most significantly, they change the contents of SIP message bodies, particularly the Session Description Protocol (SDP) bodies carried by the INVITE, 183 and 200 messages, in order to alter IP address, ports and codecs associated with proposed media streams; most commonly to facilitate NAT traversal, conference or transfer, or transcoding

This mismatch between the intended powers of intermediaries in the standard and the actual powers that intermediaries exercise in the field has significant implications for SIP security. Any entity that can change the IP address and ports of SDP can direct the media of real-time communications wherever they would like, which enables various denial of service or impersonation attacks. It also permits any intermediary to send media through a simple relay for recording or distribution. Entities that can modify SDP can moreover insert themselves as men-in-the-middle for any key exchange conducted through SDP to secure media via SRTP.

In order to detect an attack where a man-in-the-middle modified SDP, RFC3261 provided services for body-level security with S/MIME signatures. This requires a PKI and a key discovery mechanism, neither of which are trivial to provide for a protocol like SIP. Even if keying were resolved, however, operators deploying SIP so frequently relied on intermediaries modifying SDP that signing the body of SIP requests was simply infeasible. The attack of a man-in-the-middle could not be distinguished from the legitimate behavior of an operator's intermediary.

III. SIP's Security Stalemate

This has led to a decade long stalemate in rolling out SIP security. Attempts were made to try to find less unilateral ways for intermediaries to modify SIP requests (such as the "session policy" proposals), but they met with little adoption as they increased brittleness, required multiple RTTs, and increased message size to potentially unacceptable levels. Proposals to provide partial signatures over message bodies met with similar complexity concerns, though perhaps some more promising approaches are under consideration at the moment. In the interim, SIP is used in many more private network deployments, secured by nothing but walled gardens, than over the public Internet.

At this juncture, our ability to modify the most successful SIP deployments is limited. Many such SIP deployments are now more than a decade old. New features and development energy are currently focused on RTCWeb, and most changes to the SIP standard today are for maintenance purposes. As RTCWeb has good marketplace prospects, perhaps it would be better to ask what we can learn from SIP's experience that would be relevant to RTCWeb and future protocols.

SIP certainly could have been designed in a way that reduced the ability of intermediaries to work themselves into the path of SIP messages and thus reduced the potential for eavesdropping on sessions which SIP creates—and thus its usefulness as a tool of pervasive surveillance. For example, SIP allows servers to either proxy requests or to send a redirection message back to the endpoint. Redirection almost always leads to better prospects for end-to-end security than proxying. While it incurs an extra RTT, that is surely a small price to pay for security.

Had the IETF made these design decisions, however, it is unclear that SIP would have been such a successful protocol. Ultimately, many of the operators who have made SIP successful favor certain architectural models which entail that network intermediaries have significant controls over communications, architectures more reminiscent of the public switched telephone network than the canonical Internet. Operators in this space want the power to create dynamic new services in the network, rather than relying on endpoint updates to create features. Those operators are comfortable operating private networks, and are accustomed to regulators requiring law enforcement access to voice communications—they see this as a feature they can provide. The vendors who sell to those operators are a significant constituency among the participants in the IETF who work on SIP. Consequently, the participants in our consensus process want protocols like SIP to be modifiable by intermediaries for numerous reasons. This is an inherent dimension of the threat model of pervasive surveillance.

If we had designed SIP to be a protocol that didn't meet the community's requirements, it is unlikely SIP would have been such a success, and we wouldn't now be worried about protecting it from mass surveillance. But **SIP's susceptibility to mass surveillance was essential to its success**. Although I personally argued strongly at the time for greater security in SIP, in retrospect, I recognize that my arguments would have severely curtailed SIP's marketplace prospects if they had prevailed (whether that would have been good or bad is another, more complicated question). Proposed extensions to SIP that would have thwarted mass surveillance have typically failed to pass the IETF's consensus process. We could see that as a flaw, but that is indeed the purpose of a consensus process, to reflect the needs of the likely implementation and deployment community. While we could change our process so that it overrides consensus on some of these crucial points, that would result in nothing but the severing of IETF work from the reality of deployment.

IV. RTCWeb and SIP

Does RTCWeb suffer from the same problem? For the most part, no. The constituencies behind RTCWeb are radically different than those who drove SIP. The interests of browser vendors in implementing real-time communications are very different from the enablement of traditional telephone-style voice communication – no browser vendor is concerned about selling implementations to traditional telephone network operators, say. Interoperating with legacy voice services, including SIP, does remain an issue for RTCWeb, and it is the source of many of the corner cases that have made reaching consensus in RTCWeb so difficult. It is perhaps RTCWeb's greatest challenge to address those cases in a way that preserves the disruptive potential of personal communications in the web space.

Moreover, RTCWeb should be understood as part of a larger development paradigm in which Internet services push implementation code to endpoints on-the-fly. This code invokes APIs in the endpoint browser, but ultimately does so at the behest of a web service which defines the user experience. The web service takes the place of the rendezvous function of SIP, effectively tunneling SDP offer/answer messages between browsers in order to allow the browsers to form peer-to-peer connections (or at worst, to relay though TURN-like intermediaries). In many respects, RTCWeb services are thus more powerful intermediaries than those deployed by SIP operators, enabling web services far more power to instantiate features in the network.

Securing RTCWeb thus has its fair share of difficulties, not all of which are resolved today. The default expectation of RTCWeb is that the web service itself mediates authentication, which means that users must transitively trust that the service has connected them to the right party. The concept of an identity provider (rescorla-rtcweb-generic-idp) who signs the SDP offer/answer exchanges generated by the browser may allow a more direct end-to-end assurance. Trusting keying material without that assurance is inherently problematic.

Ultimately, what we learn from RTCWeb's experience in deploying security services could be applicable to SIP. Whether or not introducing these changes to SIP at this point would impact the larger SIP deployments we would want to protect from mass surveillance would remain a serious concern.

References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[3] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

[4] E. Rescorla, draft-rescorla-rtcweb-generic-idp, work in progress.