Improving Security on the Internet (with Increased Cooperation)

Hannes Tschofenig (hannes.tschofenig@arm.com)

Abstract

Securing the Internet has been an on-going activity since the early days of the IETF and a variety of technical specifications have been standardized. Someone reading through IETF RFCs might easily get the impression that the Internet should be very secure. This is, however, not the entire story as the never-ending series of breaches and recently the Snowden revelations have illustrated. While on paper everything looks pretty good many problems can be found in implementations and with deployments.

In this position paper the author makes the argument that improving the collaboration between different communities in the Internet software development life-cycle will be crucial for improving security on the Internet.

I. Introduction

The IETF has produced a variety of security specifications, including Transport Layer Security, IP security, and many application layer security mechanisms. The list of developed technologies includes end-to-end security mechanisms. The recent reports about pervasive monitoring of Internet traffic have, however, surprised many since the scale was not envisaged during the design of many Internet protocols. The revelations illustrated that implementation problems and many operational practices leave end users vulnerable regardless of a rather full toolbox of great security technologies developed by the IETF. While some of the attacks are fairly sophisticated the bulk has been exploiting rather basic security vulnerabilities; those are the type of attacks that can also be witnessed regularly in data breaches.

The author argues that existing security technologies need to get implemented correctly and deployed. While this seems to be an obvious observation various communities involved in the software-development cycle[1] are, unfortunately, not coordinating enough. In the IETF, at least, there has often been the impression that proper implementations and subsequent deployments will just happen somehow. Only infrequently efforts to influence deployment by a significant part of the IETF community are made. The most prominent example is IPv6.

To illustrate the complexity of the coordination task Section II briefly describes a simplified version of the Internet software development life-cycle. In the real world there are lots of feedback-loops and the sequence of standardization, implementation, and deployment may occur in a different order. Despite the simplification the underlying message remains the same: there are different steps in the life-cycle of Internet software development and the communities involved in each of these steps are largely different.

II. The Internet Software Life-Cycle

Developing Internet security mechanisms is a complex process that spans over a longer period of time and involves different communities. The list below illustrates the main stages of the process and highlights security problems at these different stages. In discussions about Internet security the author noticed that those with a non-computer science background often assume that Internet software development is a monolithic process and consequently demand actions from the wrong constituency.

It is worth noting that the NSA has utilized mostly known vulnerabilities on the Internet; the same vulnerabilities are exploited by criminals in data breaches every day as well.

A. Cryptographic Primitives

Internet security relies on sound cryptographic primitives, such as hash functions, random number generators, encryption algorithms, etc. These primitives form the foundation of secure communication over the Internet and serve as building blocks of security protocols, such as TLS or IPsec. Cryptanalysis aims to break these security systems and there have been various advances by researchers (e.g., with ciphers like RC4, see [8], [1]).

These cryptographic primitives are typically developed by researchers. So far, the National Institute of Standards and Technology (NIST) has spearheaded the standardization of these cryptographic primitives and has published a number of broader security guidelines in their 'Special Publications (800 Series)'. Many companies and standards developing organizations, including the IETF, have followed these recommendations.

Position paper for the 'W3C/IAB workshop on Strengthening the Internet Against Pervasive Monitoring (STRINT)', 28 February 1 March 2014, London, UK. This paper represents the views of the author and does not reflect the consensus of the Internet Engineering Task Force (IETF), of any single IETF working group, or the views of the Internet Architecture Board (IAB).

[1]The term 'software development cycle is used to emphasize the role of software. The term 'technology development cycle' would, however, also be a good fit.

B. Protocols and Architecture

Communication on the Internet relies on a great extend on standardized protocols (e.g., HTTP used for the Web platform, TLS for securing Web transactions, and POP3/IMAP/SMTP for email communication). The cryptographic primitives are building blocks in those communication protocols. Underlying these protocols is an architecture that defines the different actors to enable communication at the scale of the Internet. An architecture combines a number of individual building block into something larger. The Web 2.0 architecture, for example, not only utilizes HTTP but also on HTML, uses JavaScript, TLS for security, etc.

Industry trends have led to a more server-centric design where data is frequently uploaded to a centralized server infrastructure (instead of following a peer-to-peer paradigm). Interconnecting different applications on the Web has become the norm in so-called mash-ups. This has, however, simplified surveillance since data can be collected in a large scale from a small number of companies.

In the design of some communication architectures, such as VoIP or even email, little attention has been paid to avoid the easy collection of meta-data. Meta-data, in comparison to the actual communication content, provides information about when, where and with whom an interaction takes places. It turns out that this meta-data in itself provides incredible value for intelligence agencies (and for businesses as well).

In many cases end users are left with either uploading their personal data to servers or to not use the application at all. It is likely that this trend will continue since businesses see a lot of value in the big data analysis, which requires massive collection of data. Unfortunately, databases full of personal data provide an attractive target for intelligence agencies as well.

C. Implementations

Once the protocols and the architectures have been developed the specification needs to be turned into code. Developers typically glue various technical specifications together and thereby make choices with significant importance for security. Producing high-quality code is difficult and requires skilled programmers, sufficient penetration and interoperability testing, as well as a process and resources for quickly reacting to bug reports and discovered vulnerabilities.

Unfortunately, implementations show a number of weaknesses, such as lack of security features (see WhatsApp security [3]), include security vulnerabilities (as the OWASP Top 10 security vulnerabilities of Web applications illustrate [10]), weak pseudo-random number generators [7], and even Trojans in hardware before those get shipped to customers [9], [2]). Back-doors can also be added to software, particularly since many product implementations are not available for public review as open source code [13].

D. Deployment

Finally, once a product or service has been implemented it can be deployed, for example via a smart phone application or a Web service. Many important design decisions have to be made during this phase that impact privacy and security properties. For example, an email provider can decide about the location of their server infrastructure, whether confidentiality protection will be made available for the entire communication, how strong authentication has to be, what hardware and software platform to use, etc.

In many cases, service providers have cooperated with intelligence agencies or were forced to hand out keying material. As an example, the case of the email provider Lavabit illustrates the difficulty for companies where the owner decided to shutdown the company after being forced by the NSA to hand-over the privacy key of their Web server.

The security practices of companies have also made it easy to spy on the data of end users. As an example, when company-internal communication is not cryptographically protected problems may occur [6], [4]. Weak security of products allowed networks to be compromised [15].

III. Example

As Section II demonstrates the range of problems is fairly long and the author mainly focused on what contribution can be made by the IETF community. To illustrate some of the practical problems TLS has been chosen as an example since it has far more visibility than many other security technologies.

In Feb. 2002 the TLS working group started their standardization activities with TLS 1.1. The work was finalized in April 2006 with the publication of RFC 4346. As often in the IETF the last stages of the publication process take several months and the wider Internet security community should already be aware of the pending finalization of the work. From an IETF point of view that work is ”finished” at this point in time. Then, it is up to the implementers to take action and to add code to their TLS stacks. Due to the complexity of TLS there are a small number of open source TLS stacks that are widely used for Internet and Web communication. There are a few other closed-TLS implementations and various trimmed-down versions tailored to the embedded system environment. Interestingly, it took till 2010 till the implementation work for one of the most popular TLS stacks, the OpenSSL implementation, started. In 2012 the implementation effort with TLS 1.1 was finally concluded in OpenSSL. Consequently, there has been a gap of more than 3 years between the publication of the RFC and the start of the

implementation effort. In November 2013 the deployment status of TLS 1.1 on Web servers reached 18.2 % where TLS 1.2 already reached 20.7 %.

TLS is of course only a small part of a complete security solution for an application. Just offering TLS is by no means sufficient, particularly since the TLS protocol delegates the authorization decision to the application protocol designer. It is not uncommon (particularly in the smart phone app environment, see [5]), to find TLS being used but without certificate verification introducing man-in-the-middle attacks.

An outsider to the standardization process would assume that there is a fairly close cooperation between the standardization community and the implementation community (particularly when the eco-system is so small as in TLS with the most popular TLS stacks). Still, it takes a very long time to implement extensions or new protocol versions let alone deploy them widely. When a security vulnerability has been discovered it may take years before many products and service include the updated code. As an example, today many TLS deployments use insecure cryptographic algorithms, like RC4. With protocols that require more coordinated efforts by a large number of parties (as it is the case with DNSSEC and DANE) the situation is far more complex.

IV. Looking Forward

The description in Section II has hopefully convinced every reader that standardization is only one part of bigger Internet security puzzle. Nobody feels responsible for the entire chain and this leaves many problems unsolved. The author therefore suggests to take a step forward to increase the cooperation between the different communities.

How would the ideal state look like? First, the delay it takes from finishing a standard to the availability of implementations has to be reduced. This requires more awareness of the ecosystem and the players who are active in implementing the protocol. Second, a solid open source reference implementation has to be available. This implementation has to be complete rather than implementing only a small subset, which is often the case with implementations developed by researchers during the working group process. If there is more than one reference implementation (for example because the implementations are available in different programming languages) then those need to be visible on the IETF working group page. This reference implementation could be made available via the IETF infrastructure to allow interoperability testing. IETF participants have sporadically and on their own tried to explore online interoperability testing [2]. Additionally, it might be useful to list available implementations (in addition to the reference implementation, particularly if they are available as open source code) at a prominent place (such as the IETF working group page). This helps developers to quickly find an appropriate library. Third, those who deploy libraries need to be aware of the benefits the new features, how to configure those libraries appropriately and how to integrate them into applications. Since many application protocols use IETF security protocols the size of the eco-system and the involved partners grows considerably. Since the audience for such information is much larger (such as the IT industry in general) it might be necessary to develop education material and articles that can be tailored to the different communities, and regions. Often this education is already available although hard to find, and often incomplete or incorrect. In many cases the tutorials or articles are not written by those who are involved in the standardization process leading to second guessing about the intended purpose. IETF RFCs typically do not serve as a good tutorial for the deployment community since they were written for a different audience, namely for implementers. In case of important developments (such as a common implementation mistake or common deployment misconception) those parties should be made aware of necessary actions (similarly to security advisories but with substantially more background information). Many security protocols unfortunately suffer from a collective action problem with the consequence that unilateral actions are insufficient and raise the need for more coordinated efforts. An example of an ongoing coordination attempt can be seen in the XMPP community when the XMPP Standards Foundation (XSF) initiated a call for action to improve XMPP security [12]. The IM Observatory [14] provides information about the level of security offered by the different XMPP service providers. Since many service providers are subscribed to respective XSF hosted mailing lists they could be reached to take part in this coordinated effort.

When you look at the ideal state ask yourself how well those conditions are met with many of the security technologies developed in the IETF, such as ABFAB, OAuth, JOSE, VoIP identity management, email security, TLS, IPsec/IKEv2, DNSSEC, the RADIUS/Diameter, etc. While it is hard to imagine that all these actions would be taken by the IETF itself (as a standards organization), it is still possible to brainstorm how to the wider IETF community could, with a little bit of coordination, improve the current status quo and have a stronger cooperation. The author believes that there is a lot of room for the IETF community to improve Internet security by reaching out to other communities [3]. This additional cooperation will lead to a better feedback loop, increasing awareness of solutions, more involvement of developers, and stronger guidance from the IETF.

As a final note, every technology has its own eco-system and unique challenges. Hence, the story-line for reaching out to the respective community will therefore vary as well, as highlighted in an earlier IAB workshop by Andrei Robachevsky [11].

[2]Participants in the OAuth working group have investigated automatic, skripted interoperability testing, which is rather challenging with a multi-party protocol. With the results vendors can test their implementation at any time and let the tools play the other parties of the OAuth protocol.

[3]The Internet Architecture Board (IAB), as part of their charter, regularly interacts with other standards organizations to make each other aware of new and progress with ongoing work. Given the time that is spent interacting with some organizations the author is wondering whether that time would not be better spent to talk to some of the more practical security groups.

REFERENCES

[1] N.J. AlFardan, D.J. Bernstein, K.G. Paterson, B. Poettering, and J.C.N. Schuldt. On the Security of RC4 in TLS. In *In USENIX Security Symposium 2013*, August 2013. https://www.usenix.org/conference/usenixsecurity13/security-rc4-tls.

[2] Georg T. Becker, Francesco Regazzoni, Christof Paar, and Wayne P. Burleson. Stealthy dopant-level hardware trojans. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 197–214. Springer, 2013.

[3] Fileperms. WhatsApp is broken, really broken, September 2013. URL: http://fileperms.org/whatsapp-is-broken-really-broken/.

[4] B. Gellman and A. Soltani. NSA infiltrates links to Yahoo, Google data centers worldwide, Snowden documents say, October 2013. URL: http://www.washingtonpost.com/world/national-security/nsa-infiltrates-links-to-yahoo-google-data-centers-worldwide-snowden-documents-say/2013/10/30/e51d661e-4166-11e3-8b74-d89d714ca4dd_story_1.html.

[5] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: validating ssl certificates in non-browser software. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM Conference on Computer and Communications Security*, pages 38–49. ACM, 2012.

[6] G. Greenwald and E. MacAskill. NSA Prism program taps in to user data of Apple, Google and others, June 2013. URL: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data.

[7] Nadja Heninger. New research: There's no need to panic over factorable keys just mind your Ps and Qs, February 2012. URL: https://freedom-to-tinker.com/blog/nadiah/new-research-theres-no-need-panic-over-factorable-keys-just-mind-your-ps-and-qs/.

[8] T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii. Full Plaintext Recovery Attack on Broadcast RC4. In *Proc. the 20th International Workshop on Fast Software Encryption (FSE 2013)*, January 2013. http://home.hiroshima-u.ac.jp/ohigashi/rc4/.

[9] Jeff Larson. Revealed: The NSA's Secret Campaign to Crack, Undermine Internet Security, September 2013. URL: http://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption.

[10] OWASP. The Open Web Application Security Project (OWASP) Top Ten Project, 2013. URL: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[11] A. Robachevsky. Resilience of the commons: Routing security. In *IAB Workshop on Internet Technology Adoption and Transition (ITAT)*, December 2013.

[12] P. Saint-Andre. Xmpp ubiquitous encryption a manifesto, November 2013.

[13] New York Times. Secret Documents Reveal N.S.A. Campaign Against Encryption, September 2013. URL: http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html.

[14] XSF. Im observatory - testing the security of the jabber/xmpp network, 2013.

[15] Kim Zetter. NSA Laughs at PCs, Prefers Hacking Routers and Switches, September 2013. URL: http://www.wired.com/threatlevel/2013/09/nsa-router-hacking/.