Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

C. Cramers
L. Garratt
University of Oxford
N. Sullivan
Cloudflare
October 30, 2017

Randomness Improvements for Security Protocols
draft-sullivan-randomness-improvements-00

Abstract

Randomness is a crucial ingredient for TLS and related transport security protocols. Weak or predictable cryptographically-strong pseudorandom number generators (CSPRNGs) can be abused or exploited for malicious purposes. See the Dual EC random number backdoor for a relevant example of this problem. This document describes a way for security protocol participants to mix their long-term private key into the entropy pool from which random values are derived. This may help mitigate problems that stem from broken CSPRNGs.

Table of Contents

1. Introduction

Randomness is a crucial ingredient for TLS and related transport security protocols. TLS in particular uses Random Number Generators (RNGs) to generate several values: session IDs, ephemeral key shares, and ClientHello and ServerHello random values. RNG failures such as the Debian bug described in [DebianBug] can lead to insecure TLS connections. RNGs may also be intentionally weakened to cause harm [DualEC]. In such cases where RNGs are poorly implemented or insecure, an adversary may be able to predict its output and recover secret Diffie-Hellman key shares that protect the connection.

This document proposes an improvement to randomness generation in security protocols inspired by the "NAXOS trick" [NAXOS]. Specifically, instead of using raw entropy where needed, e.g., in generating ephemeral key shares, a party's long-term private key is mixed into the entropy pool. In the NAXOS key exchange protocol, raw entropy output x is replaced by H(x, sk), where sk is the sender's private key. Unfortunately, as private keys are often isolated in HSMs, direct access to compute H(x, sk) is impossible. An alternate but functionally equivalent construction is needed.

The approach described herein replaces the NAXOS hash with the keyed hash, or PRF, wherein the key is derived from raw entropy output and a private key signature.

2. Randomness Wrapper

Let x be the raw entropy output of a CSPRNG. When properly instantiated, x should be indistinguishable from a random string of length |x|. However, as previously discussed, this is always true. To mitigate this problem, we propose an approach for wrapping the

CSPRNG output with a construction that artificially injects randomness into a value that may be lacking entropy.

Let PRF(k, m) be a cryptographic pseudorandom function, e.g., HMAC [RFC2104], that takes as input a key k of length L and message m and produces an output of length M. For example, when using HMAC with SHA256, L and M are 256 bits. Let Sig(sk, m) be a function that computes a signature of message m given private key sk. Let G be an algorithm that generates random numbers from raw entropy, i.e., the output of a CSPRNG. Let tag be a fixed, context-dependent string. Lastly, let KDF be a key derivation function, e.g., HKDF-Extract [RFC5869], that extracts a key of length L suitable for cryptographic use.

The construction is simple: instead of using x when randomness is needed, use:

PRF(KDF(G(x) || Sig(sk, tag)), tag)

Functionally, this computes the PRF of a fixed string with a key derived from the CSPRNG output and signature over the fixed string. The PRF behaves like a truly random function from 2^L to 2^M assuming the key is selected at random. Thus, the security of this construction depends on secrecy of Sig(sk, tag) and G(x). If both are leaked, then the security reduces to the scenario wherein this wrapping construction is not applied.

In systems where signature computations are not cheap, these values may be precomputed in anticipation of future randomness requests. This is possible since the construction depends solely upon the CSPRNG output and private key.

3. Application to TLS

The PRF randomness wrapper can be applied to any protocol wherein a party has a long-term private key and also generates randomness. This is true of most TLS servers. Thus, to apply this construction to TLS, one simply replaces the "private" PRNG, i.e., the PRNG that generates private values, such as key shares, with:

HMAC(HKDF-Extract(nil, G(x) || Sig(sk, tag)), tag)

Moreover, we fix tag as "TLS 1.3 Additional Entropy" for TLS 1.3. Older variants use similarly constructed strings.

4. IANA Considerations

This document makes no request to IANA.

5. Security Considerations

A security analysis was performed by two authors of this document. Generally speaking, security depends on keeping the private key secret. If this secret is compromised, the scheme reduces to the scenario wherein the PRF random wrapper was not applied in the first place.

6. Normative References

[DebianBug] Yilek, Scott, et al, ., "When private keys are public - Results from the 2008 Debian OpenSSL vulnerability", n.d., <https://pdfs.semanticscholar.org/fcf9/fe0946c20e936b507c023bbf89160cc995b9.pdf>.

[DualEC] Bernstein, Daniel et al, ., "Dual EC - A standardized back door", n.d., <https://projectbullrun.org/dual-ec/documents/dual-ec-20150731.pdf>.

[NAXOS] LaMacchia, Brian et al, ., "Stronger Security of Authenticated Key Exchange", n.d., <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/strongake-submitted.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

Authors' Addresses

Cas Cremers
University of Oxford
Wolfson Building, Parks Road
Oxford
England

Email: cas.cremers@cs.ox.ac.uk

Luke Garratt
University of Oxford
Wolfson Building, Parks Road
Oxford
England

Email: luke.garratt@wolfson.ox.ac.uk

Nick Sullivan
Cloudflare
101 Townsend St
San Francisco
United States of America

Email: nick@cloudflare.com