Tinjauan Kode dan Penilaian Keamanan
Untuk
Lelantus Spark
Pengiriman Awal: 9 Desember 2022
Pengiriman Akhir: 19 Desember 2022

Disiapkan untuk
Reuben Yap | Firo
Levon Petrosyan | Firo
Aaron Feickert | Cypher Stack
Aram Jivanyan| Firo
Peter Shugalev | Firo

Disiapkan oleh
Mikerah Quintyne-Collins | HashCloak Inc Manish
Kumar | | HashCloak Inc
Onur Inanc Dogruyol | HashCloak Inc

Daftar Isi

Ringkasan Eksekutif

Komunitas Firo melibatkan HashCloak Inc untuk melakukan audit terhadap implementasi protokol Lelantus Spark yang merupakan protokol privasi generasi berikutnya dari Firo. Lelantus Spark memberikan peningkatan pada konstruksi Lelantus asli dan fitur yang lebih ramah pengguna. Audit dilakukan dari tanggal 24 Oktober 2022 hingga 28 November 2022.

Kami diberikan komitmen 88b499355c8781fad279ba3171ca8938bf0907ae dan perubahan minimal dilakukan saat kami menjalani audit yang tidak mempengaruhi ruang lingkup audit. Cakupan audit adalah file-file yang disertakan dalam folder `libspark` dan `spark.`

Peninjauan basis kode dilakukan setelah tim pengembangan Firo membaca laporan tersebut. PR #1218 dan Commit 60dba45a8aee17cae0bc24d12d4071b39583781b ditinjau ulang untuk perbaikan yang kami sarankan. Selanjutnya, beberapa temuan dinilai kembali setelah klarifikasi oleh tim pengembangan Firo.

Selama audit berlangsung, kami terutama berfokus pada temuan masalah yang dapat menyebabkan hal-hal berikut ini:

- Pengeluaran ganda
- Mencetak koin baru secara gratis
- Mendanonimkan transaksi
- Jumlah transaksi yang bocor
- Bocornya kunci pengeluaran
- Pembelanjaan yang tidak sah tidak dapat dilakukan. Pengeluaran yang diotorisasi berarti pengguna memiliki kunci pengeluaran
- Tidak memungkinkan untuk menghubungkan transaksi satu sama lain atau ke alamat (yang beragam)
- Tidak memungkinkan untuk menautkan alamat yang beragam

Secara keseluruhan, kami menemukan masalah yang berkisar dari yang sedang hingga yang bersifat informasi. Kami menemukan bahwa kode dalam ruang lingkup mengikuti makalah dengan sangat cermat dalam deskripsi protokolnya. Khususnya, untuk konstruksi kriptografi, implementasinya mengikuti dengan sangat dekat dengan deskripsi makalah terkait. Penjelasan ini mengarah pada kemudahan pemeliharaan dan auditabilitas basis kode.

Tingkat Keparahan	Jumlah Temuan
Kritis	0
Tinggi	0
Sedang	1
Rendah	5
Informasi	2

Ikhtisar

Lelantus Spark adalah protokol privasi generasi berikutnya dari Firo. Protokol ini mencakup berbagai peningkatan pada konstruksi Lelantus seperti privasi penerima dan banyak lagi
jaminan keamanan yang beralasan. Selain itu, ini mencakup beberapa fitur kegunaan seperti pembuktian yang didelegasikan, pengungkapan selektif, dan multisignature. Kriptografi
primitif dan konstruksi yang digunakan dalam implementasi Lelantus Spark dipahami dengan baik dan bergantung pada asumsi kriptografi yang umum. Selain itu, mereka juga digunakan dalam koin privasi yang sebanding seperti Monero.

Cakupan

Ruang lingkup audit ini adalah sebagai berikut:

- Semua berkas di dalam `src/libspark`
- Semua file di dalam `src/spark`

Jika ada ketergantungan langsung dalam basis kode Firo yang digunakan dalam dua folder di atas yang kami yakini berdampak pada keamanan audit,
Kami telah memasukan kedalam ruang lingkup audit secara minimal. Secara khusus, kami juga mempertimbangkan hal-hal berikut:

- `src/secp256k1`
- `src/crypto`

sesuai kebutuhan untuk lebih memahami dan menilai keamanan kode dalam ruang lingkup.

Temuan

FLS-01: Alokasi memori tidak ada untuk A_scalars

Jenis: Rendah
File yang terpengaruh: src/libspark/bpplus.cpp

Deskripsi: Pada `BPPlus::prove`, pada baris 139 & 140, `A_points.reserve(2*N*M+1);` dideklarasikan dua kali untuk `A_points` tetapi tidak ada alokasi memori untuk `A_scalars.`

Dampak: Realokasi memori berulang kali dapat terjadi pada `A_scalars` yang dapat mempengaruhi efisiensi dan juga dapat mengakibatkan kebocoran memori. Kebocoran memori sangat kecil kemungkinannya karena realokasi vektor terjadi secara otomatis ketika vektor tersebut keluar dari ruang lingkup.

Saran: baris 140 harus diganti dengan `A_scalars.reserve(2*N*M+1);`

Status: Setelah berdiskusi dengan tim pengembang, skor penilaian temuan ini diturunkan dari Medium ke Low. Kesalahan ketik ini telah diperbaiki pada komit 5dd4cc7154bd0e139b2e4d76da9529f4643a5715.

FLS-02: Pengecekan yang hilang jika tantangan sama dengan 0 di berbagai tempat

Jenis: Sedang
File yang terpengaruh: src/libspark/bpplus.cpp

Deskripsi: Di berbagai tempat, setelah transkrip diperbarui, tantangan dihitung dengan melakukan hashing pada transkrip dan elemen bukti yang relevan. Namun, tantangan tidak diperiksa jika sama dengan 0 dalam Skalar.

Dampak: Jika sebuah tantangan sama dengan 0, maka prover jahat dapat membalikkan 0, merusak kesehatan protokol. Secara efektif akan direduksi menjadi pemasangan hash serangan preimage.

Saran: Hal ini dapat diperbaiki dengan pemeriksaan dan mengulang komputasi jika pemeriksaan gagal. Tuliskan kasus uji untuk NOL di baris 7

`const Scalar ZERO = Scalar((uint64_t) 0);`

```
// In Line 155, 387
Scalar y = transcript.challenge(“y”);

if(y == ZERO) {
  MINFO(“y is 0, try again”);
  goto try_again;
}

// In Line 156, 395
Scalar z = transcript.challenge(“z”);

if(z == ZERO) {
  MINFO(“z is 0, try again”);
  goto try_again;
}

// In Line 255, 403
Scalar e = transcript.challenge(“e”);

if(e == ZERO){
  MINFO(“e is 0, try again”);
  goto try_again;
}

// In Line 286, 409
Scalar e1 = transcript.challenge(“e”);

if(e1 == ZERO){
  MINFO(“e is 0, try again”);
  goto try_again;
}
```

Status: Pengecekan nol ini telah ditambahkan hanya pada `BPPlus::verify`. Ini tidak ditambahkan ke `BPPlus::prove` karena agar penyerang dapat menerapkan kondisi ini dari sudut pandang p e m b u k t i a n , mereka hanya akan mengabaikan pemeriksaan sama sekali dan dapat membalikkan 0. Oleh karena itu, cukup dengan menambahkan pemeriksaan ini ke

untuk memastikan bahwa verifier menangkap prover yang berbahaya. Pemeriksaan telah ditambahkan pada komit 5dd4cc7154bd0e139b2e4d76da9529f4643a5715.

FLS-03: Periksa kehilangan untuk vektor komitmen pedersen ganda

Jenis: Rendah
File yang terpengaruh: src/libspark/grootle.cpp

Deskripsi: Pada fungsi `vector_commit` di baris 65, validitas dari pernyataan komitmen pedersen ganda tidak diperiksa, yaitu apakah ukuran vektor `Gi` sama dengan `a` dan `Hi` sama dengan `b.`

Dampak: Memberikan argumen yang salah ke `vector_commit` dapat menyebabkan kesalahan segmentasi.

Saran: Seharusnya ada pemeriksaan untuk pernyataan pedersen.

```
if (Gi.size() != a.size() || Hi.size() != b.size()) {
     Throw std::invalid_argument(" Bad pedersen statement");
}
```

Status: Pemeriksaan ini telah diimplementasikan pada commit 5dd4cc7154bd0e139b2e4d76da9529f4643a5715

FLS-04: Menggunakan data rahasia untuk batas perulangan

Jenis: Rendah
File yang terpengaruh: src/libspark/keys.cpp

Deskripsi: Dalam keys.cpp di dalam `GetHex()`, perulangan pada baris 187 secara langsung bergantung pada `buffer`. `buffer` tidak secara langsung bergantung pada data rahasia tetapi bergantung pada `Q1` dan `Q2` pada baris 179 dan 180 secara berurutan.
Dampak: Loop dengan bound yang berasal dari nilai rahasia secara langsung mengekspos program ke serangan waktu

Saran:

```
// In Line 187, instead of using the following

for (const auto b : buffer) {
    ss << (b >> 4);
    ss << (b & 0xF);
}

// First use the following, then use the loop

size_t buffer_size = buffer.size();
```

Status: Batas perulangan yang terikat tidak bergantung pada data rahasia dan karenanya perubahan ini tidak dilakukan. Tetapi, `GetHex()` dan `SetHex()` sudah ketinggalan zaman dan oleh karena itu kedua fungsi tersebut telah dihapus pada komit [60dba45a8aee17cae0bc24d12d4071b39583781b](60dba45a8aee17cae0bc24d12d4071b39583781b).

FLS-05: Nilai pergeseran perlu diketik untuk meminimalkan perilaku yang tidak terdefinisi

Jenis: Rendah
File yang terpengaruh: src/libspark/bech32.cpp

Keterangan: Jika tidak diketik, operator shift kiri bitwise mungkin tidak terdefinisi dan dapat menyebabkan kesalahan runtime.

Dampak: Dapat menyebabkan kesalahan saat runtime.

Saran: Di Baris 116

```
uint8_t c0 = (uint8_t) c >> 25;
```

Status: Setelah berdiskusi dengan tim pengembang, ditemukan bahwa perubahan ini merusak kemampuan beach32 untuk mendeteksi perubahan alamat tertentu. Oleh karena itu, perubahan ini tidak dilakukan.

FLS-06: Fungsi Randomize( ) harus dapat mengembalikan nol

Jenis: Rendah
File yang terpengaruh: secp256k1/src/Scalar.cpp

Deskripsi: Fungsi `randomize()` yang digunakan di src/libspark dan src/spark untuk menghasilkan skalar acak tidak menghasilkan skalar `nol`. Khususnya, baris 220 pada Scalar.cpp adalah salah.

Dampak: Mengecualikan nol dalam `randomize() akan` menyebabkan fungsi pengacakan yang bias, di mana distribusi dari semua angka mungkin tidak memiliki kemungkinan yang sama.

Saran: Baris 220 harus diganti dengan

```
while(!this->isMember());
```

Status: Masalah ini telah diperbaiki pada commit 60dba45a8aee17cae0bc24d12d4071b39583781b

FLS-07: Pengoptimalan dalam verifikasi Chaum

Jenis: Informasi
File yang terpengaruh: src/libspark/chaum.cpp

Deskripsi: Pada baris 141, pengoptimalan dapat dilakukan pada penjumlahan A2 dengan menambahkan `proof.A2[i]` terlebih dahulu dan kemudian memanggil `points.emplace_back` pada penjumlahannya. Dengan cara ini, ruang skalar n-1 dan ruang titik n-1 dapat dihemat.

Saran:

```
GroupElement A2_sum;
A2_sum = proof.A2[0];
for(std::size_t i = 1; i < n; i++) {
```

```
    A2_sum += proof.A2[i];
}
scalars.emplace_back(w);
points.emplace_back(A2_sum);
```

Status: Saran ini berguna untuk efisiensi dan karenanya perlu ditambahkan pada commit 5dd4cc7154bd0e139b2e4d76da9529f4643a5715

FLS-08: Hasil operasi kurva/kelompok perlu diperiksa ulang agar valid

Jenis: Informasi
File yang terpengaruh: src/libspark/bpplus.cpp, src/libspark/chaum.cpp, src/libspark/grootle.cpp, src/libspark/schnorr.cpp

Deskripsi: Di berbagai tempat, operasi kurva/kelompok digunakan. Hasil operasi ini perlu diperiksa ulang untuk memastikan bahwa hasilnya valid.

Dampak:

Saran: Pada file yang ditentukan di atas, hasilnya perlu diperiksa. Fungsi-fungsi berikut ini perlu disisipkan setelah operasi kurva/kelompok.

```
// In src/secp256k1/include/GroupElement.h, the following functions
// can be used

isMember();
isInfinity();

// In src/secp256k1/include/Scalar.h, the following function  // can
be used

isMember();
```

di src/libspark/bpplus.cpp

```
// In Line 151

proof.A = A_multiexp.get_multiple();

// In Line 248, 249

L_ = L_multiexp.get_multiple();
R_ = R_multiexp.get_mutliple();

// In Line 517

Return multiexp.get_multiple().isInfinity();

// In Line 260, 261

Gi1[i] = Gi1[i] * e_inverse + Gi1[i+N1] * (e*y_N1_inverse);
Hi1[i] = Hi1[i] * e + Hi1[i+N1]*e_inverse;

// In Line 281, 282

proof.A1 = Gi1[0]*r_ + Hi1[0]*s_ + G*(r_*y*b1[0] + s_*y*a1[0]) +
H*d_;

proof.B = G*(r_*y*s_) + H*eta_;


In Line 288, 289, 290

proof.r1 = r_ + a1[0]*e1;
proof.s1 = s_ + b1[0]*e1;
proof.d1 = eta_ + d_*e1 + alpha*e1.square();
```

Di src/libspark/chaum.cpp

```
// In Line 62
```

```
proof.A1 = H*t;

// In Line 65, 66

proof.A1 += F*r[i] + G*s[i];
proof.A2[i] = T[i]*r[i] + G*s[i];

// In Line 72, 75, 76, 77, 78

proof.t3 = t;
proof.t1[i] = r[i] + c_power*x[i];
proof.t2 += s[i] + c_power*y[i];
proof.t3 += c_power*z[i];
c_power *= c;
```

Di src/libspark/schnorr.cpp

```
// In Line 43

proof.A = G*r;

// In Line 50, 51

proof.t += y[i].negate()*c_power;
c_power *= c;

// In Line 78
c_power *= c;
```

Status: Setelah berdiskusi dengan tim pengembang, disimpulkan bahwa pemeriksaan penutupan ini harus dilakukan di tingkat pembungkus kurva/skalar dan bukan di `libspark.` oleh karena itu, pemeriksaan penutupan terpisah di `libspark` tidak diperlukan.