Report 1907551

Source Code Review
Proton Crypto Library

for
Proton Technologies AG

conducted by
SEC Consult

Version: 1.2  |  Date: 2019-05-14
Responsible: SEC Consult  |  Author: SEC Consult
Confidentiality class: Public
Table of Contents

1. Management Summary ........................................................................................................... 3
   1.1 Scope and Timetable ........................................................................................................... 3
   1.2 Results ............................................................................................................................... 3
   1.3 Disclaimer .......................................................................................................................... 4

2. Vulnerability Summary ......................................................................................................... 5
   2.1 Total Risk Per System ......................................................................................................... 5
   2.2 Risk of Each Vulnerability ................................................................................................. 5

3. Detailed Analysis .................................................................................................................... 6
   3.1 Proton Crypto Library ......................................................................................................... 6
      3.1.1 General Information ...................................................................................................... 6
      3.1.2 Cleartext message spoofing - FIXED ........................................................................... 6
      3.1.3 Small RSA keys are allowed - FIXED .......................................................................... 8
      3.1.1 Notes ............................................................................................................................ 9

4. Appendix ................................................................................................................................ 10
   4.1 sig_spoof.go ...................................................................................................................... 10

5. Version History ....................................................................................................................... 13
1 Management Summary

The following chapter summarizes the scope and timetable of the code review, the results of the code review, and outlines the measures recommended by SEC Consult.

1.1 Scope and Timetable

During the initial security assessment for Proton Technologies AG, SEC Consult performed a source code review of the Proton Crypto Library - a set of supplementary cryptography libraries written in Go, which allows encrypting, decrypting, signing, and verifying messages using symmetric and asymmetric cryptography. Objective of the review was to reveal security issues and to offer suggestions for improvement. The focus of the code review was to identify:

- vulnerabilities in cryptographic algorithms,
- routines that could cause user data compromise,
- routines that could be abused for user monitoring.

The initial review was conducted in March 2019 and a total effort of 7 days was dedicated to identifying and documenting security issues in the code base of the Proton Crypto Library.

The following subsystems were in the scope of the code review:

- openpgp
- ed25519
- brainpool

The source code of the library was publicly available and could be obtained from the respective online repository (https://github.com/ProtonMail/crypto/tree/5bcbe4637f9ffaa5c899e9a5c1f2bcdac5b881a9).

In April 2019, Proton Technologies fixed the identified issues and supplied the fixes to SEC Consult for verification (https://github.com/ProtonMail/crypto/tree/efb430e7512f00d8d9aebdb254fc14ef76954880). Goal of the fix verification was to confirm remediation provided by the applied fixes. SEC Consult verified the fixes in May 2019.

1.2 Results

During the initial code review, SEC Consult found a high-risk vulnerability in the reviewed source code. An attacker can, given that specific preconditions are met, abuse this vulnerability to:

- spoof arbitrary cleartext messages without invalidating the signatures.

In addition, SEC Consult discovered a medium-risk vulnerability in the reviewed source code. An attacker can, given that specific preconditions are met, abuse these vulnerabilities to:

- force victims to generate small-size private keys.

All security issues that were identified in the initial code review were properly fixed by Proton Technologies AG.
1.3 Disclaimer

At the request of Proton Technology AG, this report has been declassified from strictly confidential to public. While the report was shortened for public release, relevant vulnerability information has been maintained.

In this particular project, a timebox approach was used to define the consulting effort. This means that SEC Consult allotted a prearranged amount of time to identify and document vulnerabilities. Because of this, there is no guarantee that the project has discovered all possible vulnerabilities and risks.

Furthermore, the security check is only an immediate evaluation of the situation at the time the check was performed. An evaluation of future security levels or possible future risks or vulnerabilities may not be derived from it.
2 Vulnerability Summary

This chapter contains all identified vulnerabilities in the reviewed source code of the company Proton Technologies AG.

<table>
<thead>
<tr>
<th>Risk assessment</th>
<th>Initial no. of vulnerability classes</th>
<th>Current no. of vulnerability classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Medium</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>High</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Critical</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

2.1 Total Risk Per System

The following table contains a risk assessment for each system which contained security flaws.

<table>
<thead>
<tr>
<th>System</th>
<th>Field of application</th>
<th>Initial risk</th>
<th>Current risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proton Crypto Library</td>
<td>Cryptography Library</td>
<td>High</td>
<td>-</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>High</td>
<td>-</td>
</tr>
</tbody>
</table>

2.2 Risk of Each Vulnerability

The following table contains a risk assessment for the discovered vulnerabilities.

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>System</th>
<th>Initial risk</th>
<th>Current risk</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleartext message spoofing</td>
<td>Proton Crypto Library</td>
<td>High</td>
<td>FIXED</td>
<td>6</td>
</tr>
<tr>
<td>Small RSA keys are allowed</td>
<td>Proton Crypto Library</td>
<td>Medium</td>
<td>FIXED</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>High</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
3 Detailed Analysis

This chapter outlines the attacks and found vulnerabilities in detail.

3.1 Proton Crypto Library

3.1.1 General Information

This section describes vulnerabilities found in the Proton Crypto Library. The parts of the library, which were reviewed implement the core OpenPGP functionalities as well as key generation, encrypting, decrypting, signing and signature verification functionalities.

3.1.2 Cleartext message spoofing - FIXED

According to RFC 4880 Chapter 7, the cleartext signed message can contain one or more optional “Hash” Armor Headers. The “Hash” Armor Header specifies the message digest algorithm(s) used for the signature. However, the Proton Crypto library ignores the value of this Header, which allows an attacker to spoof it. Thereby, an attacker can lead a victim to believe the signature was generated using a different message digest algorithm than what was actually used.

Moreover, since the library skips Armor Header parsing in general, an attacker cannot only embed arbitrary Armor Headers, but also prepend arbitrary text to cleartext messages without invalidating the signatures.

CVSS-v3 Base Score: 7.1 (High)

This issue also affects mainline Go Cryptography Libraries from which Proton Crypto Library was forked. SEC Consult contacted the respective vendor and a patch has been published for this library as well by now.

3.1.2.1 Recheck results

Like in the initial code review, for the recheck the following cleartext messages were used:

- the cleartext message with a valid SHA-1 signature;
- the message with a valid SHA-1 signature was tampered by changing the value of the “Hash” Armor Header from SHA-1 to SHA-512;
- the message with a valid SHA-1 signature was modified by embedding Unicode-encoded “LINE TABULATION” in the “Hash” Armor Header.
The script `sig_spoof.go` performing signature verification is given in section 4.1. The output of the script shows, that the issue was successfully fixed:

```
$ go run sig_spoof.go
Verifying not tampered...
Signature accepted!

Verifying spoofed hash...
failed to check signature: %s openpgp: invalid data: unknown hash algorithm in cleartext message headers
Verifying spoofed cleartext...
No cleartext text found
```

For protecting against the initial vulnerability, a new, higher-level API `clearsign.Block.VerifySignature`, was added. It is recommended to use this method instead (marked as bold text in `sig_spoof.go` script in section 4.1).
3.1.3 Small RSA keys are allowed - **FIXED**

Several parameters can be configured externally where the size of the generated RSA key can be specified. However, the key generation routine does not perform checks on the given key size (except if equal to zero). Thus, if an attacker can influence the external configuration, the small-sized key can be generated, which can be used for compromising the legitimate user.

CVSS-v3 Base Score: 5.8 (Medium)


3.1.3.1 Recheck results

The following code fragment (same as in the initial code review) was used to verify if the issue is fixed:

```go
package main

import ("fmt" "golang.org/x/crypto/rsa" "golang.org/x/crypto/openpgp" "golang.org/x/crypto/openpgp/packet")

func main() {
    if err != nil {
        fmt.Println(err)
    }
}
```

The output of the script shows, that the issue was successfully fixed:

```bash
$ go run poc_short_keys.go
crypto/rsa: bits must be >= 1024
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x8 pc=0x51bc2f]

goroutine 1 [running]:
main.main()
    /go/src/proton/short_rsa_keys/poc_short_keys.go:14 +0xdf
exit status 2
```
3.1.1 Notes

The following lists peculiarities and minor issues that were identified during the code review.

1. Implementation allows both interpretation and generation of RSA Sign-Only and RSA Encrypt-Only keys. The following code fragment from *openpgp/packet/private_key.go* demonstrates the issue:

```go
func NewSignerPrivateKey(currentTime time.Time, signer crypto.Signer) *PrivateKey {
    pk := new(PrivateKey)
    switch pubkey := signer.Public().(type) {
    case rsa.PublicKey:
        pk.PublicKey = *NewRSAPublicKey(currentTime, &pubkey)
        pk.PubKeyAlgo = PubKeyAlgoRSASignOnly
    case ecdsa.PublicKey:
        pk.PubKeyAlgo = PubKeyAlgoRSAEncryptOnly
    }
    return pk
}
```

According to RFC 4880 Chapter 9.1 this is deprecated and should not be used anymore: “RSA Encrypt-Only (2) and RSA Sign-Only are deprecated and SHOULD NOT be generated but may be interpreted.”

It is recommended to use key flags instead.

Fix verification on 2019-04-30: the issue was fixed.
4 Appendix

4.1 sig_spoof.go

```go
package main

import (
    "fmt"
    "golang.org/x/crypto/openpgp/clearsign"
    "golang.org/x/crypto/openpgp"
    "golang.org/x/crypto/openpgp/packet"
    "bytes"
)

func verify(input []byte) {
    var err error
    b, _ := clearsign.Decode(input)
    if b == nil {
        fmt.Println("No clearsign text found")
        return
    }
    keyring, err := openpgp.ReadArmoredKeyRing(bytes.NewBufferString(signingKey))
    if err != nil {
        fmt.Println(err)
        return
    }
        fmt.Println(err)
        return
    }
    b, _ = clearsign.Decode(input)
        fmt.Println("failed to check signature: %s", err)
        return
    }
    fmt.Println("Signature accepted!
")
}

func main() {
    fmt.Println("Verifying not tampered...")
    verify(no_spoof)
    fmt.Println("Verifying spoofed hash...")
}
```
verify(hash_spoof)
    fmt.Println("Verifying spoofed cleartext...")
verify(cleartext_spoof)
}

var no_spoof = []byte(`
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Message to be signed
-----BEGIN PGP SIGNATURE-----
iQEzBAEBAgAdFiEAAXWUn665cAXgInLZXVs62dBO+u4FAlyeCMMACgkQXVs62dBO
+u6WeQqAvOTZakwXCZ2woIBhK+g3fgOiCOF8YtXgZCyDYZgR/JIf1+iCh7lWajq
9/JcniN79lX6hxy4qoT8loLAHNeoUzSkKiliRmcQPtfcPiNRCRtAnKDFkiA5N
0C9CesJYXoASBraFugxeI7q29tVdPNC8WvjTja72yafu4b63TXkdcu+TCHtH51v
10rq5jEJT/+UGyc0+gbvegsAoNhmp8q9kJmST6kJgmCs9Tj1AmeX1wTV55f5L+
7pR452BmlA7o141vYwv+G1KJygrPzeQokybF2rFRuMxjlvqfo1/4LrtXgA/7
v8H32sgUV9T/HNx5bFPQjJbOhBVRg==
=bb6N
-----END PGP SIGNATURE-----``)

var hash_spoof = []byte(`
------BEGIN PGP SIGNED MESSAGE------
Hash: SHA1024
Message to be signed
------BEGIN PGP SIGNATURE------
iQEzBAEBAgAdFiEAAXWUn665cAXgInLZXVs62dBO+u4FAlyeCMMACgkQXVs62dBO
+u6WeQqAvOTZakwXCZ2woIBhK+g3fgOiCOF8YtXgZCyDYZgR/JIf1+iCh7lWajq
9/JcniN79lX6hxy4qoT8loLAHNeoUzSkKiliRmcQPtfcPiNRCRtAnKDFkiA5N
0C9CesJYXoASBraFugxeI7q29tVdPNC8WvjTja72yafu4b63TXkdcu+TCHtH51v
10rq5jEJT/+UGyc0+gbvegsAoNhmp8q9kJmST6kJgmCs9Tj1AmeX1wTV55f5L+
7pR452BmlA7o141vYwv+G1KJygrPzeQokybF2rFRuMxjlvqfo1/4LrtXgA/7
v8H32sgUV9T/HNx5bFPQjJbOhBVRg==
=bb6N
------END PGP SIGNATURE------``)

var cleartext_spoof = []byte(`
------BEGIN PGP SIGNED MESSAGE------`
"\nHash: SHA512\u000b This data is part of the header"
`
Message to be signed
# 5 Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Status/Changes</th>
<th>Created by</th>
<th>Responsible</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2019-03-29</td>
<td>Initial report</td>
<td>SEC Consult</td>
<td>SEC Consult</td>
</tr>
<tr>
<td>1.1</td>
<td>2019-05-02</td>
<td>Fix verification</td>
<td>SEC Consult</td>
<td>SEC Consult</td>
</tr>
<tr>
<td>1.2</td>
<td>2019-05-14</td>
<td>Public report</td>
<td>SEC Consult</td>
<td>SEC Consult</td>
</tr>
</tbody>
</table>