fritz vs. GnuPG

Gründe für eine Neuentwicklung
Februar 2013

Pro GnuPG

- OpenPGP-Standard nach RFC 4880
- Open Source (GNU-GPL)
- patentfrei
- Windows, Linux, Mac
- große Verbreitung
- Web of Trust (Vertrauensnetzwerk: Bob signiert den Schlüssel von Alice und spricht so sein Vertrauen aus) c't + cacert

Contra GnuPG

- erstes Release Ende 1997 > alte Standard's (RSA, AES, HMAC)
- Hauptentwickler unfähig bzw. unwillig, bestehende Sicherheitsprobleme zu beseitigen
- halbe Sicherheit?
  - http://blog.fefe.de/?ts=bb558613 (Mon Jan 15 2007) "Werner Kochs stinkenden Gnupg-Sourcen nach Bugs zu gucken" "Ich habe geguckt. Einmal in gnupg 1.4.6 und einmal in gnupg 2.0.1" "Dann kann gnupg halt von mir aus schlecht bleiben."

- "Im Übrigen fände ich das alles nur halb so erbärmlich, wenn Werner nicht öffentliche Förderkohle abgegriffen hätte, um seine Softwareentwicklung zu finanzieren."
- http://blog.fefe.de/?ts=bb5547d2 (Mon Jan 15 2007)

  Bitte beachtet:

  - Ich behaupte nicht, daß alle Teile des Patches exploitbare Sicherheitsprobleme fixen.
  - Ich bin nur ein Mensch. Ich hab da wahrscheinlich Dinge übersehen, oder vielleicht macht mein Patch auch was schlimmer. ...

- Ich habe mich auf Schreibzugriffe beschränkt. Es gibt da immer noch zig Stellen, wo gnupg wild in der Gegend rumliest. Wenn das nicht wie ein Krypto-Schlüssel Info Leak aussah, hab ich es in Ruhe gelassen. Gnupg ranzt eh an diversen Stellen mit assert() oder BUG() ab, wenn was nicht stimmt, insofern scheint ein spontanes Abrauchen für gnupg-Standards akzeptables Verhalten zu sein.
- Das ist auch der Grund, wieso ich bei meinen Fixed häufig auf assert() zurück gegriffen habe.

- Und jetzt bitte ich darum, daß ihr das nicht einfach alle benutzt, sondern euch selber die Sourcen auch noch mal anguckt. Die Leute erzählen immer alle, wie toll sicher freie Software sei, weil so viele Augen draufgucken — dann müssen aber auch viele Augen draufgucken!!
- Update: Und schon hat jemand (Fabian Keil, danke!) einen Fuckup gefunden. Neues Diff (Jan 15 17:20) geuploaded.
- Patch besteht aus 500 Zeilen
- Der Patch: http://dl.fefe.de/gnupg.dif

Kurzer Blick in den GnuPG-Patch

Zeile 10 - 21

```
      /* Calculate new size of the area and allocate */
      n0 = oldarea? oldarea->len : 0;
-     n = n0 + nlen + 1 + buflen; /* length, type, buffer
*/
+
+//     n = n0 + nlen + 1 + buflen; /* length, type,
buffer*/
+
+     n = n0 + nlen;
+     assert(n > n0);
+     assert(n + buflen > n); // abort if assertion is
false
+     n += buflen;
+     assert(n + 1 > n);
+     ++ n;
```

Was ist ein Integer Overflow?

- Wir addieren zwei 32bit Zahlen, was erhalten wir? > eine Zahl mit 33bit (Überlauf)
- der CPU verarbeitet aber nur 32bit > falsches Ergebnis
  - Beispiel: `0xffffffff + 1 = 0x100000000 = 0`
  - Lösung: `if(a+23<23) overflow`

Wir brauchen sichere Alternativen > http://nacl.cr.yp.to

- Entwickelt von Daniel J. Bernstein (*djb*) (University of Illinois at Chicago), Tanja Lange (Technische Universität Eindhoven), und Peter Schwabe (Academia Sinica)
- kurze Einführung in NaCl: http://cr.yp.to/highspeed/coolnacl-20120725.pdf

Vorteile von NaCl

- Crypto Kram wird in der Lib erledigt
  - Beispiel aus *cryptlib*:

```
cryptCreateEnvelope(&cryptEnvelope,cryptUser,
CRYPT_FORMAT_SMIME );
cryptSetAttributeString(cryptEnvelope,
CRYPT_ENVINFO_RECIPIENT,recipientName, recipientNameLength
);
cryptPushData( cryptEnvelope,message,messageSize,&bytesIn
);
cryptFlushData( cryptEnvelope );
cryptPopData( cryptEnvelope,encryptedMessage,
encryptedSize, &bytesOut );
cryptDestroyEnvelope( cryptEnvelope );
```

---

gleiches macht NaCl mit folgendem Aufruf:

```
c = crypto_box(m,n,pk,sk);
```

- immun gegen alle derzeit bekannten Angriffe (Cache-Timing-Attacken)
- keine dynamische Speicherverwaltung in der Lib
  - einfache Implementierung
  - viele Fehler in C in der Speicherverwaltung
  - volle Kontrolle über den Speicher `rand_mem();`
- sehr schnell
- patentfrei
- Open Source + gemeinfrei
- Portierung von NaCl nach Windows > http://njör.de/Projekte/subnacl.html

- NaCl nutzt Skalarmultiplikation mit elliptischen Kurven Curve25519, nicht RSA
- NaCl nutzt Salsa20, nicht AES
- NaCl nutzt Poly1305, nicht HMAC
- sehr kurze Schlüssel (32bytes, 44 Zeichen) <> Web of Trust (hier kann nur der Hash des Schlüssel verglichen werden, bei NaCl kann der Schlüssel direkt am Telefon ausgetauscht werden)

Pro fritz

- Portabel, static build (dll-Injection)
- Minimierung dynamischer Speicherverwaltung auf **2 Blöcke**
  - diese wurden mit `stralloc()` implementiert (sichere Stringverwaltung von djb)
- geheimer Schlüssel wird mit Kennwort und Schlüsseldatei gesichert
- kleiner Source (~2000 Zeilen, davon ~1010 Zeilen WinAPI) + Open Source

Pro fritz

- es gibt kein Protokoll > verschlüsselter Text/Dateien sind von Zufall nicht zu unterscheiden

Contra fritz

- fritz wurde für Microsoft Windows entwickelt, läuft jedoch mit wine auch auf Mac oder Linux
  - gtk+ ist keine Option (static build ohne upx ~8MB groß)

Download und mehr Info's zu fritz:

http://njör.de/Projekte/fritz.html