Passwort

Michael Roth DAC698@DBQ400.#FB.HES.DEU.EU

COLLABORATORS

	TITLE : Passwort	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Michael Roth DAC698@DBQ400.#FB.HES.DEU.EU	August 26, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Passwort **1**

1.1 Passwortgenerator 1

1.2 fwü 1

1.3 need 2

1.4 paras 2

1.5 funk 3

1.6 schad 4

1.7 autor 4

1.8 pd 4

Chapter 1

Passwort

1.1 Passwortgenerator

```
                Passwortgenerator ist FreeWare / PuplicDomain
          und wurde geschrieben von:
Michael Roth - DAC698@DBQ400.#FB.HES.DEU.EU

 (C) 1998/99 Michael Roth - Billy The Byte
```

Für was überhaupt ?

Vorraussetzungen

Parameter

Funktionserklärung

Schadensersatzansprüche

Autor

Anmerkung zu Puplic Domain

1.2 fwü

Passwort generator wurde, wie es der Name schon sagt, geschrieben um Passwörter zu erzeugen. Passwörter sind eine Sache, die bei falscher Anwendung gefährlich werden können.

Passwörter werden oftmals so ausgesucht, das man sie sich leicht merken kann. So steht oftmals Adressen etc. in dem Passwort. Eine person, die ein wenig über den Autor des Passwortes bescheid weiß, kann dies dann leicht knacken. Ich probierte dies bei jemanden mal aus. Nach dem ich mir fragmente seines Passwortes besorgte (bei PR immer 5er Packete) konnte ich nach kurzer Zeit das Passwort nahezu vollständig generieren und hatte zugang zu der BBS in der er

Arbeitet. Er verwendete seinen Call und seine Anschrift, die einzelnen Elemente durch ein / getrennt. Dies füllte er so auf 80 Zeichen auf.

Ich kenne das Problem selbst. Die Phaspharse für meinenn PGP Key ist ebenfalls aus einer bestimmten erratbaren , aber langen Zeichenfolge zusammen gesetzt. Jedoch überall da, wo ich Passwörter Automatisieren kann setze ich Zufallspasswörter ein, wenn möglich mit Binärzeichen.

Der Nachteil liegt auf der Hand: Das auswendig lernen, geschweiger denn das schreiben der Passwörter wird schwer, jedoch genauso schwer ist es das Passwort zu erraten. das einfachste wäre dann alle auszutesten. Bei 80 Zeichen langen Passwörter wäre das immerhin 256^80 :)

Nach einem längeren Kampf verstanden dies nun einige Leute und begannen sich zufallspasswörter zu generieren. Jedoch wurden sie schnell faul und änderten legedlich ein paar Zeichen eines alten Passwortes. Dies is wieder nischt, wenn ich mit nem Fremden Passwort zugang erhalte...(vorallem leicht wenn nur fragmente des Passwortes überprüft werden).

Passwortgenerator hat nun die aufgabe, dies zu umgehen, schnell und bequem genau definierte Passwörter anzulegen.

Auch ich bin nicht geweiht gegen zufällige doppelkeitserscheinungen, bei Tests trat das jedoch nicht auf.

1.3 need

Auch wenn es peinlich ist, ich kann die genaue definition nicht treffen. Ich habe STORMC vor kurzen erhalten und mich noch nicht richtig auseinander gesetzt damit.

Ich denke, das OS2.0 ausreichend ist.

Die CPU sollte mindestens mal ne 68000 sein :) Speicherverbrauch ? Dynamisch, je nach Parameter...

Getestet wurde es mit A1200 68030 OS3.1 ; geringer gehts nicht. Ich wäre Dankbar wenn jemand das mal mit OS1.3 OS2.0 testen könnte. Danke!

1.4 paras

Obwohl ReadArgs nicht verwendet wird, verwende ich die Syntax mal um die Parameter aufzuzeichnen:

Quellfile/A, Zielfile/A, Start/N, End/N, Length/N, douple/n, Offset/N

Quellfile

Weil mir der zufallsgenerator zu linear war, und das verschleiern mit der Zeit zu wenig war, entschloss ich mich noch ein zusatzfile einzubauen, das als weiere zufällge Quelle diehnt.

Zielfile

Hier wird das File angegeben, worin das Passwort gespeichert wird.

Start

Nummerische Angabe. Stellt die unterste Grenze der Zeichen dar, die für die Passwort generierung heranngezogen werden sollen. Minimum 0

End

Nummerische Angabe. Stellt die oberste Grenze dar, aus denen Passwörter bestehen sollen. Max: 256

Length

Gibt die Länge des Passwortes an. Min 1

double

Gibt an, wie oft sich ein Zeichen wiederholen darf. Hier ist zu achten, das genügen Zeichen für die länge des Passwortes übrig bleiben. So ist eine Länge von 1000 mit ner Zeichenbreite von 100 und ner wiederholungsrate von 1 nicht möglich. Berechnen lässt es sich folgend: Länge/Zeichenbreite und das auf die nächste volle Zahl gerundet. Auf PR sind 80 Zeichen üblich, die meisten Filter erlauben von CHR$(33) - CHR$(128). Wir haben 95 Zeichen. double kann also 1 Betragen.

Sollte ein Passwort dennoch mal nicht möglich sein, nach kurzer Zeit nochmal testen, da die Zeit ebenfalls für die generierung verwendet wird, un der Zufall es will das nicht genügend Zeichen zusammen kommen :) (Tja, die technik)

Offset

Wurde eingefügt, um im Zielfile unterschiedliche startpossitionen zu erreichen.

Es wäre ja möglich, das jemand immer die gleiche Zeit und das gleiche File verwendet; dann käme auch als das gleiche Passwort bei raus. Damit auch solch eine Person noch eine Chance hat, kann er den Offset flexibel gestalten.

1.5 funk

Als erstes wird irgendein File, was durch den User bestimmt wird, eingelesen. Danach wird der (lineare) zufallsgenerator angeworfen und die Systemzeit ausgelesen, die in Sekunden seit irgendwass angegeben werden. Nun haben wir 3

Ergebnisse, die mit einem Exclusiv Oder verknüpft werden. Das ergebniss wird mit den Grenzen verglichen und danach ob es bereits vorhanden ist. Wenn nein wird es in das Zielfile geschrieben.

Keine große Kunst, aber es geht.

Zu beachten ist die Zeit! Auf dem amiga gibt es Kontinuierlich die Sekunden seit 1978, also immer eine andere Zahl. Bei der Portierung muß darauf achtgegeben werden, die das betreffende System die Zeit Handhabt und ob das ergebniss verwendet werden kann.

1.6 schad

Die einzigen Schadensansprüche die gestellt werden können kommen vom Autor, und zwar dann wenn eine menge wütender User auf ihn einschlagen :)

Ich übernehme keine Garantie für die Funktion des Programmes, wenngleich es auch mehrmals getestet wurde. Es hat nicht die gewohnte Qualität wie der rest meiner Programme, da ich C noch nicht so kenne. Die Qualität meiner zukünftigen C Programme wird jedoch steigen...:)

Für schäden die druch verwendung, vorallem falscher verwedung meines Programm angerichtet werden kann ich nicht haftbar gemacht werden.

Die verwendung geschieht auf EIGENE GEFAHR!

1.7 autor

Der verantwortliche für Passwortgenerator ist:

Michael Roth
Hiesbach 4
35410 Hungen Langd

DAC698@DBQ400.#FB.HES.DEU.EU

1.8 pd

Ich hatte vor den Quelltext ebenfalls rauszuegen. Obwohl ich selbst kein so guter Programmierer bin, sah ich, wie das Programm mißbraucht wurde zur angabe, oder totaler schrott implementiert wurde, so das ich mich entschied zukünftig nurnoch FreeWare zu schreiben. Wer den Quelltext dennoch haben will, soll dies begründet sagen, und er soll ihn erhalten. Die Rechte bleiben weiterhin beim Autor!