**Dokumentation**

**Projekt Brute-Force**

*Projektarbeit Matlab Workshop*

Fachhochschule Nordwestschweiz

Studiengang Systemtechnik

Klasse 8Sbb

Dozent: Prof. Dr. Norbert Hofmann

Projektteam: Adrian Gonzalez

Bruno Hürzeler

Brugg-Windisch, Donnerstag, 24. Mai 2018, V 1.0000.00

# Inhalt

[Inhalt 2](#_Toc514925673)

[Änderungen 3](#_Toc514925674)

[Abbildungen 3](#_Toc514925675)

[Tabellen 3](#_Toc514925676)

[Formeln 3](#_Toc514925677)

[1 Aufgabenstellung 4](#_Toc514925678)

[1.1 Rahmenbedingungen 4](#_Toc514925679)

[1.2 Projektidee 4](#_Toc514925680)

[1.3 Implementierung / Tools 4](#_Toc514925681)

[1.4 Funktionsbeschreibung 5](#_Toc514925682)

[2 Programmaufbau 6](#_Toc514925683)

[3 Bedienung der Software 7](#_Toc514925684)

[3.1 Einstellungen von Matlab 7](#_Toc514925685)

[3.2 Programm starten 8](#_Toc514925686)

[3.3 System evaluieren 9](#_Toc514925687)

[3.4 Daten eintragen und Einstellungen auswählen 10](#_Toc514925688)

[3.5 Brute-Forcing ohne GPU starten 11](#_Toc514925689)

[3.6 Erfolgreiches Brute-Forcing ohne GPU 13](#_Toc514925690)

[3.7 Abbruch des Brute-Forcing 14](#_Toc514925691)

[3.8 Brute-Forcing mit der GPU 15](#_Toc514925692)

[3.9 Generieren eines Hashs 16](#_Toc514925693)

[4 Projektmanagement 20](#_Toc514925694)

[4.1 Soll – Ist Vergleich des Lastenheftes 20](#_Toc514925695)

[4.2 Commits 20](#_Toc514925696)

[4.3 Aufwand pro Funktion 20](#_Toc514925697)

[5 Abnahmetests 21](#_Toc514925698)

[5.1 Abnahmetest der Module in «functions.cpp» 21](#_Toc514925699)

[5.2 Abnahmetest der Systemanforderungen 27](#_Toc514925700)

[5.3 Abnahmetest des Gesamtsystems 30](#_Toc514925701)

[5.4 Geplante Optimierungen 32](#_Toc514925702)

[6 Ehrlichkeitserklärung 33](#_Toc514925703)

## Änderungen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Index** | **Kapitel Verweis** | **Dok. Rev.** | **Datum / Wer** | **Zweck** |
| 1 | Alle | V00.0000.00 | 17.05.18 / Hub | Erstellung des Dokumentes |
| 2 | Kapitel 2 | V00.0000.01 | 24-05.18 / Hub | Ausarbeitung des Kapitels |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Tabelle Änderungen

## Abbildungen

[Abbildung 1 Programmaufbau 6](file:///C:\DATA\FHNW\8_Semester\matl\matlProject\doc\matl_Dokumentation_Gonzalez_Huerzeler.docx#_Toc514925704)

[Abbildung 2 Matlab Einstellungen 7](file:///C:\DATA\FHNW\8_Semester\matl\matlProject\doc\matl_Dokumentation_Gonzalez_Huerzeler.docx#_Toc514925705)

[Abbildung 3 Programm starten 8](file:///C:\DATA\FHNW\8_Semester\matl\matlProject\doc\matl_Dokumentation_Gonzalez_Huerzeler.docx#_Toc514925706)

[Abbildung 4 System-Evaluierung 9](file:///C:\DATA\FHNW\8_Semester\matl\matlProject\doc\matl_Dokumentation_Gonzalez_Huerzeler.docx#_Toc514925707)

[Abbildung 5 Dateneingabe nach Evaluation 10](#_Toc514925708)

[Abbildung 6 Bereit für Brute-Forcing ohne GPU 11](#_Toc514925709)

[Abbildung 7 Nach Brute-Forcing-Start ohne GPU 12](#_Toc514925710)

[Abbildung 8 Erfolgreiches Brute-Forcing ohne GPU 13](#_Toc514925711)

[Abbildung 9 Abbrechen des Brute-Force-Vorgangs 14](#_Toc514925712)

[Abbildung 10 Aktivieren der GPU 15](#_Toc514925713)

[Abbildung 11 Hash Generator 16](#_Toc514925714)

## Tabellen

[Tabelle 1 Änderungen 3](#_Toc514925715)

[Tabelle 2 Beschreibung der Elemente 6](#_Toc514925716)

[Tabelle 3 Zeitaufwand in Stunden 20](#_Toc514925717)

## Formeln

[Formel 3: Refresh-Rate Display 20](#_Toc514337662)

# Aufgabenstellung

## Rahmenbedingungen

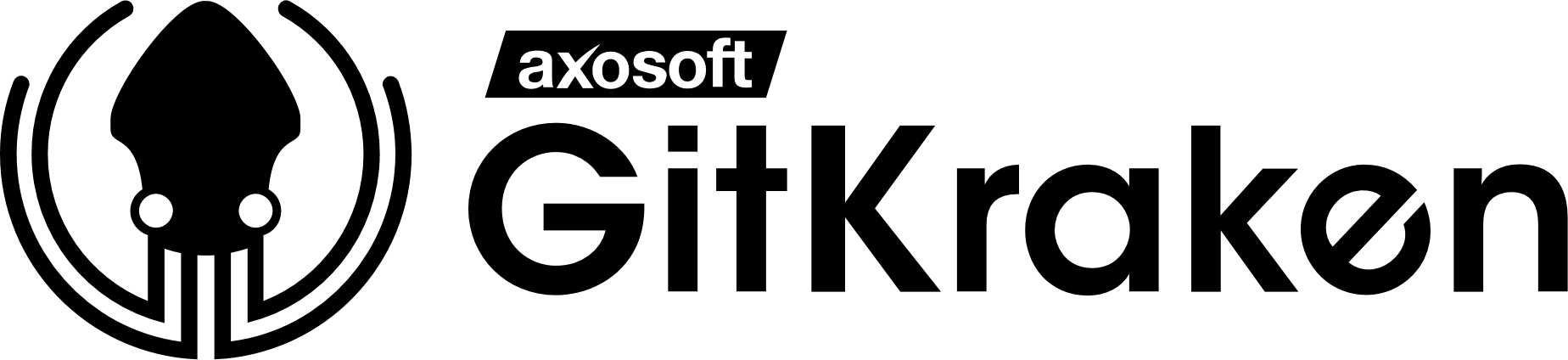
In dem Projekt sollen Daten aus Benutzereingaben, Daten oder Bildern eingelesen, gefiltert oder verändert werden. Die Ergebnisse dieser Operation werden graphisch dargestellt (Plots, Bilder, Video) und die Ergebnisse wieder auf der Festplatte abgespeichert.

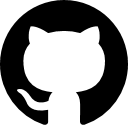
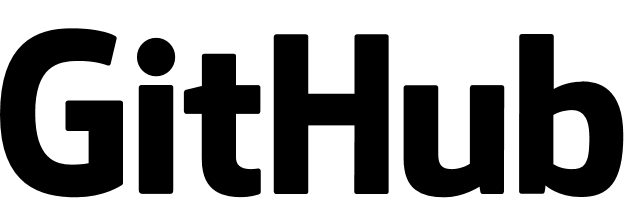
Die Arbeit wird während des Frühlingssemesters durchgeführt. Während der Unterrichtszeit des Frühlingssemesters sollten zusammen mindestens 40 Stunden investiert werden.

## Projektidee

Bla

## Implementierung / Tools

Die Implementierung erfolgt mittels folgenden Tools:

* Matlab, *Version R2017b (9.3.0.713579)*
* GitHub
* GitKraken, *Version 3.6.1*

Das GitHub wird als Versionenkontrolle verwendet, um möglichst effizient im Team arbeiten zu können. Das Projektmanagemet wurde mittels dem integrierten IssueTool «Glo» von GitKraken durchgeführt.

Das GitHub kann unter folgender URL abgerufen und ggf. heruntergeladen / geklont werden:

<https://github.com/adrn1990/matl_project.git>

## Funktionsbeschreibung

# Programmaufbau



**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**11**

**12**

**13**

**14**

**15**

**16**

**17**

**18**

**19**

**20**

Abbildung Programmaufbau

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Element** | **Sub-Element** | **Beschreibung** |
| 1 | File | *New run…* | Alle Daten bereinigen |
|  |  | *Save…* | Speichern der Daten aus dem Log-Monitor |
|  |  | *Exit* | Beenden des Programmes |
| 2 | ? | *About…* | Informationen über die Software und Entwickler |
| 3 | Evaluate System |  | Starten der Systemevaluation |
| 4 | Mode |  | Modus Auswahl |
| 5 | Input |  | Eingabefeld für das Passwort oder Hash |
| 6 | Encryption |  | Auswahl des Verschlüsselungs-Algorithmus |
| 7 | Cluster |  | Auswahl des zu verwendenden Clusters |
| 8 | GPU |  | An- und Abwählen der GPU |
| 9 | Start |  | Start Brute-Force |
| 10 | Abort |  | Abbruch eines Brute-Force Vorganges |
| 11 | Status |  | Status- Fortschrittsanzeige in Prozent |
| 12 | Log-Monitor |  | Informationsanzeige |
| 13 | CPU Graph |  | Visualisierung des Gebrauchs der CPU Leistung |
| 14 | GPU Graph |  | Visualisierung des Gebrauchs der GPU Leistung |
| 15 | CPU load |  | Anzeige des Gebrauchs der CPU Leistung |
| 16 | CPU temp |  | Anzeige der CPU Temperatur |
| 17 | GPU load |  | Anzeige des Gebrauchs der GPU Leistung |
| 18 | GPU temp |  | Anzeige der GPU Temperatur |
| 19 | Result |  | Ausgabefeld in welchem das Resultat angezeigt wird |
| 20 | Create a hash |  | Link auf eine Webseite um einen Hash zu generieren |

Tabelle Beschreibung der Elemente

# Bedienung der Software

## Einstellungen von Matlab

Abbildung Matlab Einstellungen

Um den vollen Funktionsumfang der Brute-Force Software nutzen zu können, sollte das MATLAB® aus Administrator ausgeführt werden. Die Abbildung 1 zeigt auf, wie Sie das Programm standardmässig als Administrator ausführen.

MatlabXXXXy.exe ->Rechte Maustaste -> Eigenschaften -> Erweitert -> Hacken setzen

## Programm starten

Abbildung Programm starten

Das Starten der Brute-Force Software erfolgt mittels der im Projekt enthaltenen Anwendung.

Doppelklick auf Brute Force VX.X.exe

## System evaluieren

Abbildung System-Evaluierung

Damit die Brute-Force Software funktionieren kann, muss als erstes das System evaluiert werden. Solange die Systemdaten nicht erfasst wurden, können keine anderen Bedienelemente ausgewählt werden.

## Daten eintragen und Einstellungen auswählen

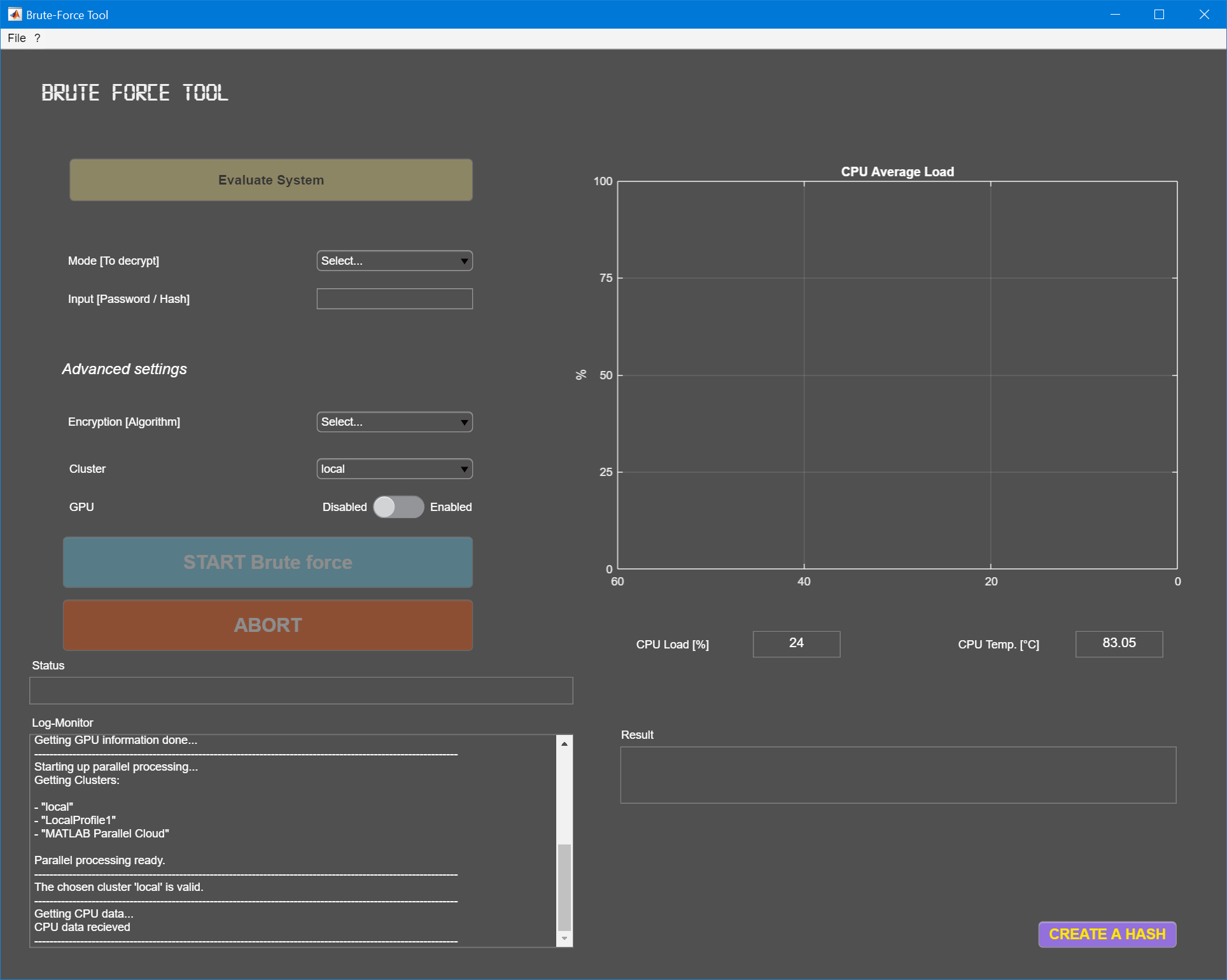


Abbildung Dateneingabe nach Evaluation

Nach erfolgreichem Evaluieren des Systems können die evaluierten Daten aus dem Log-Monitor entnommen werden. Zudem ist es nun möglich, die Daten für das Brute-Forcing einzustellen.

## Brute-Forcing ohne GPU starten

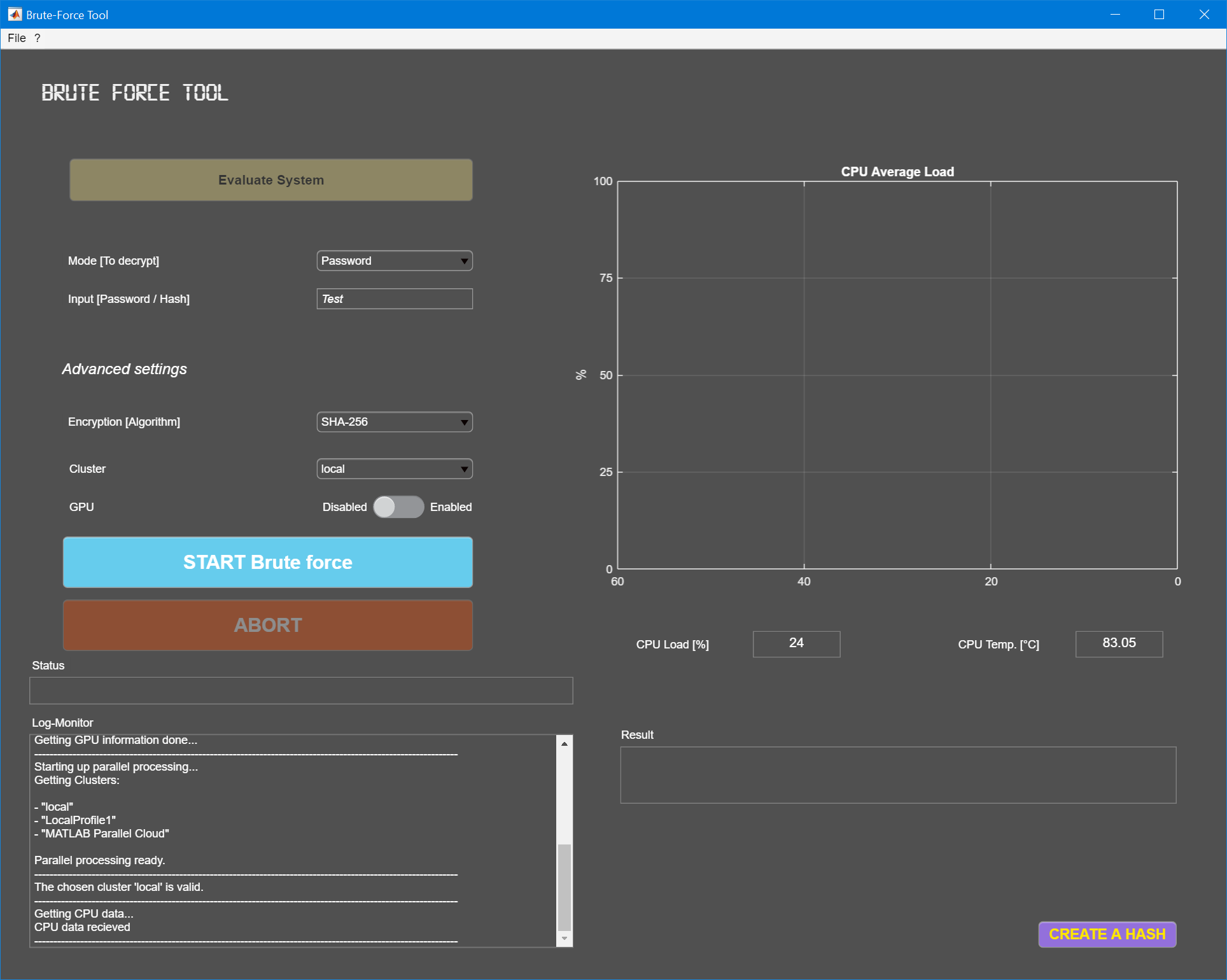


Abbildung Bereit für Brute-Forcing ohne GPU

Sobald alle erforderlichen Daten, welche für das Brute-Forcing benötigt werden, eingestellt sind wird das Start-Bedienelement aktiv und ein Brute-Forcing kann gestartet werden.

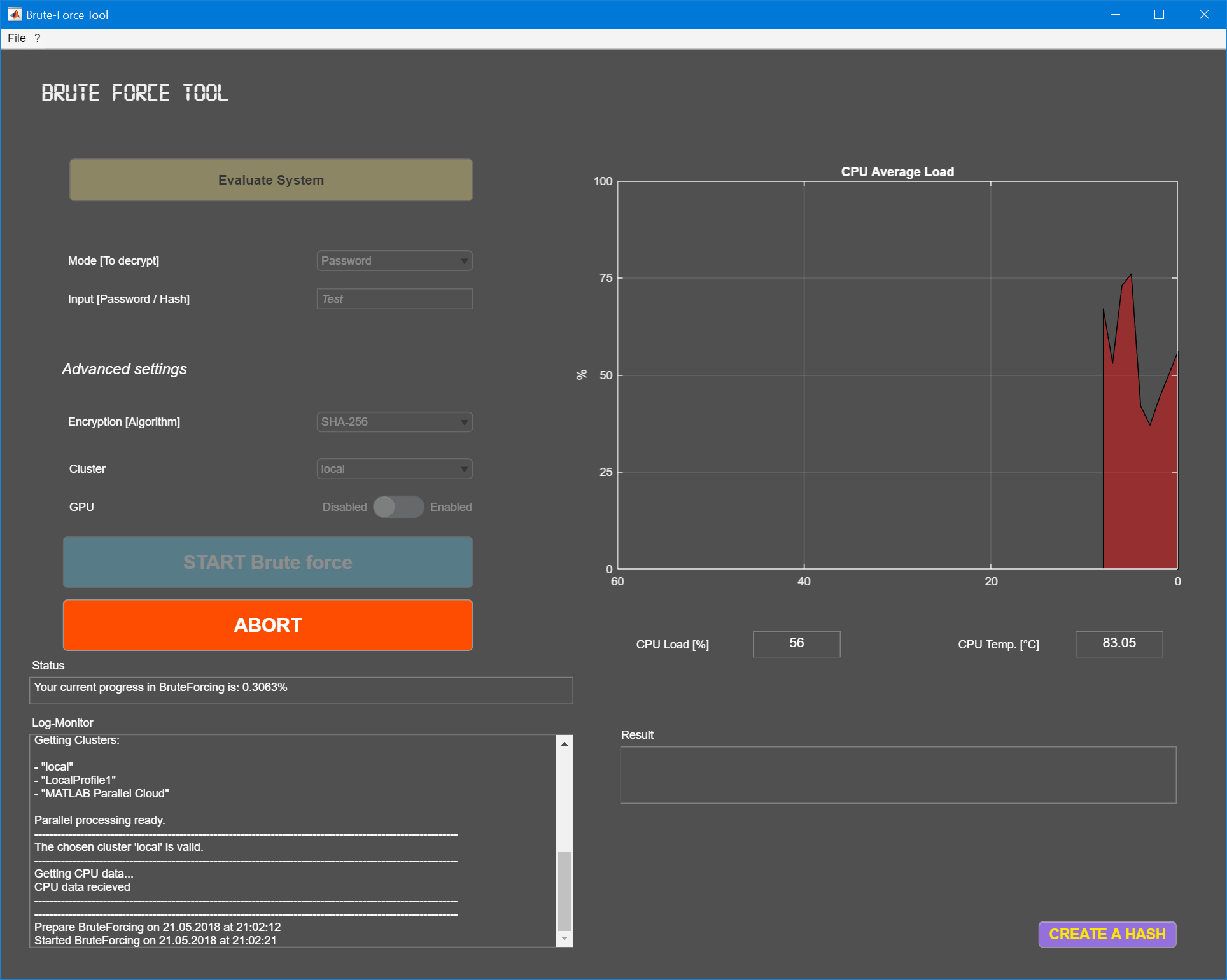


Abbildung Nach Brute-Forcing-Start ohne GPU

Aus dem Log-Monitor kann entnommen werden, wann die Software sich vorbereitet und anschliessend der Brute-Force-Prozess begonnen hat.

Weiter können aktuelle Informationen aus der Grafik oder der Statusanzeige entnommen werden.

## Erfolgreiches Brute-Forcing ohne GPU

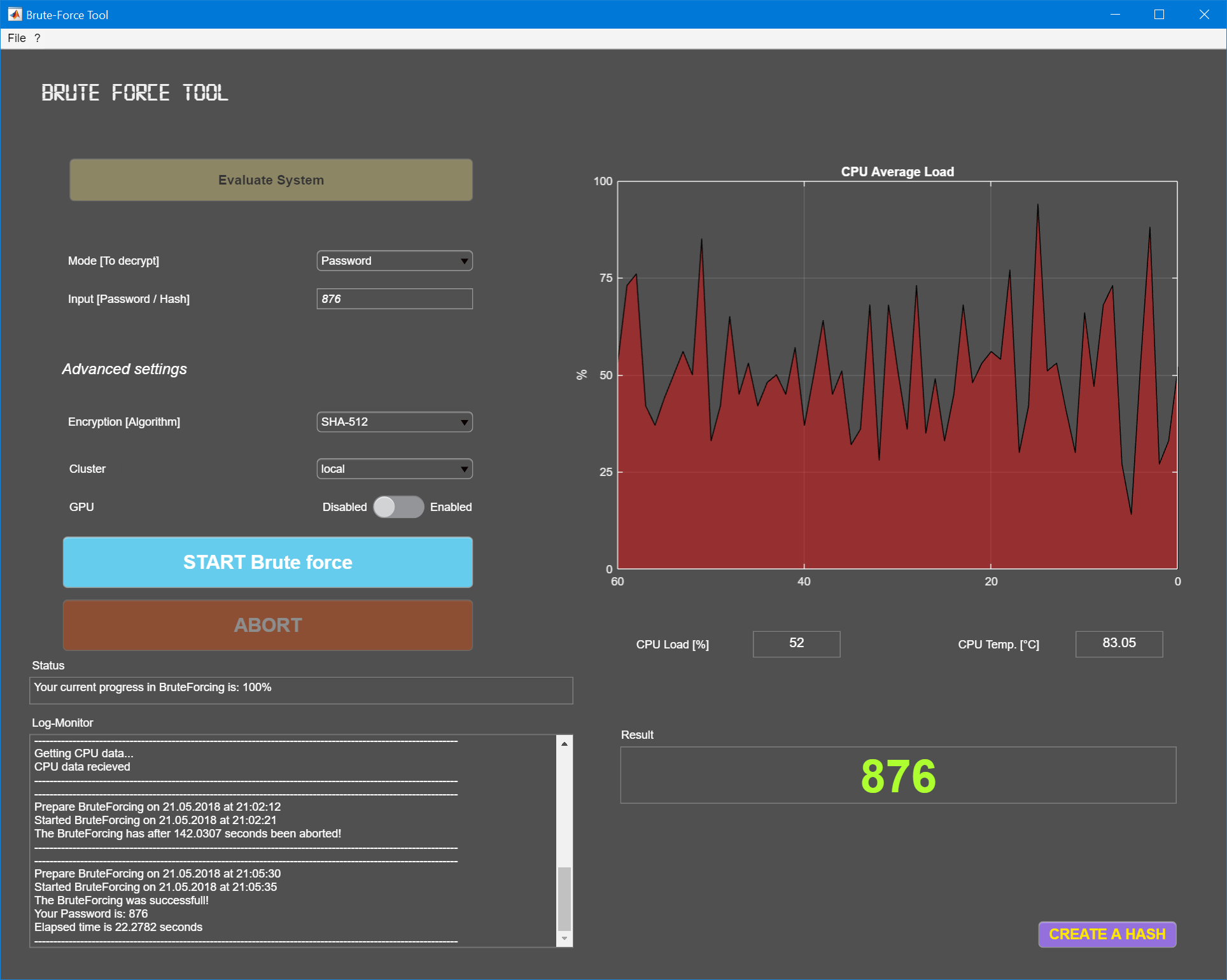


Abbildung Erfolgreiches Brute-Forcing ohne GPU

Nach erfolgreichem Brute-Forcing des Passwortes oder Hashes wird dieses im Resultat-Ausgabefeld visualisiert und ein neuer Vorgang kann anschliessend wieder gestartet werden.

## Abbruch des Brute-Forcing

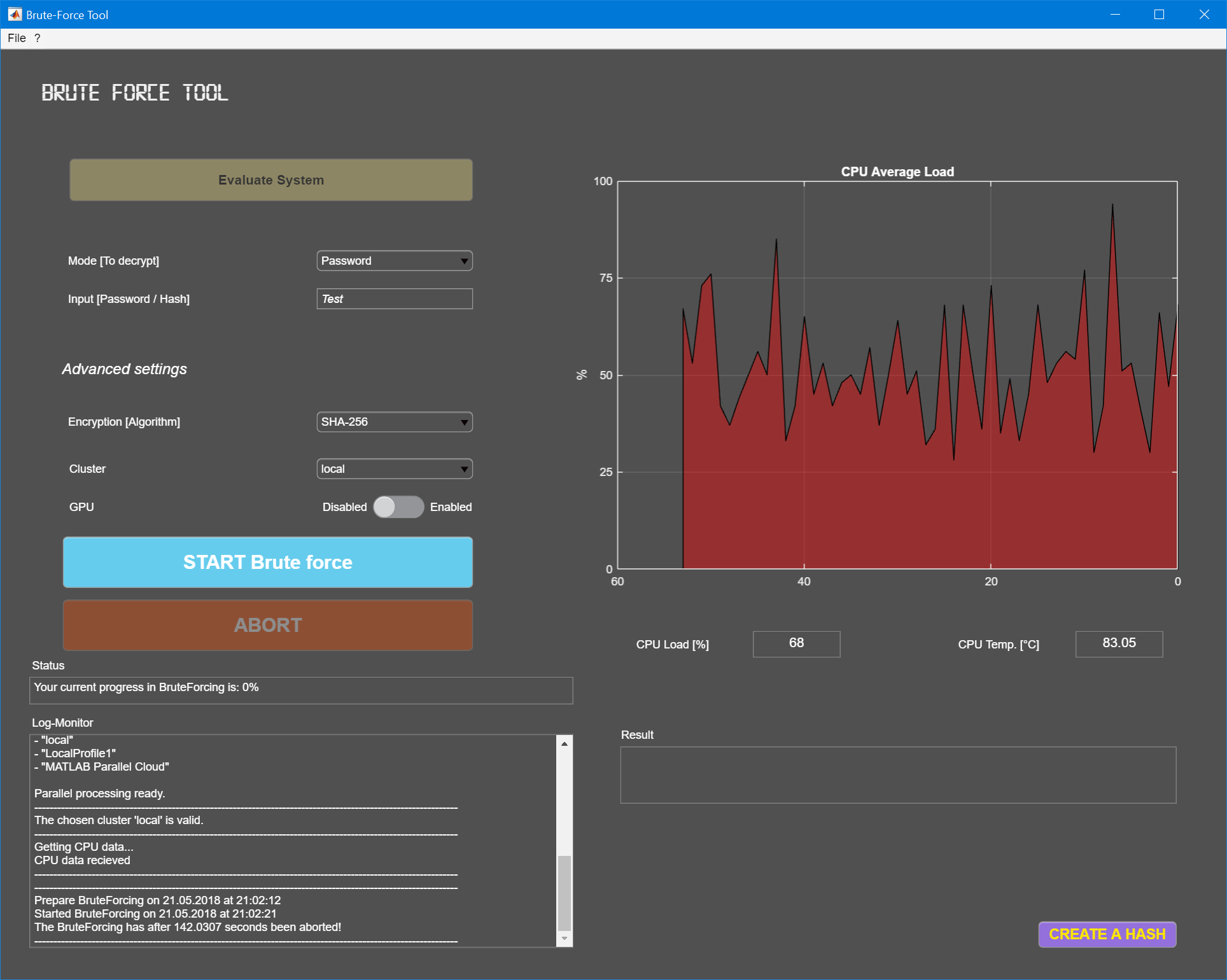


Abbildung Abbrechen des Brute-Force-Vorgangs

Für einen Abbruch des Vorganges kann dies mittels des Abort-Knopfes durchgeführt werden. Sobald der Star-Knopf aktiv wird und im Log-Monitor das Abbrechen bestätigt wurde, kann ein neuer Prozess gestartet werden.

Dieser Vorgang ist unabhängig mit welcher Hardware der Prozess gestartet wurde.

## Brute-Forcing mit der GPU

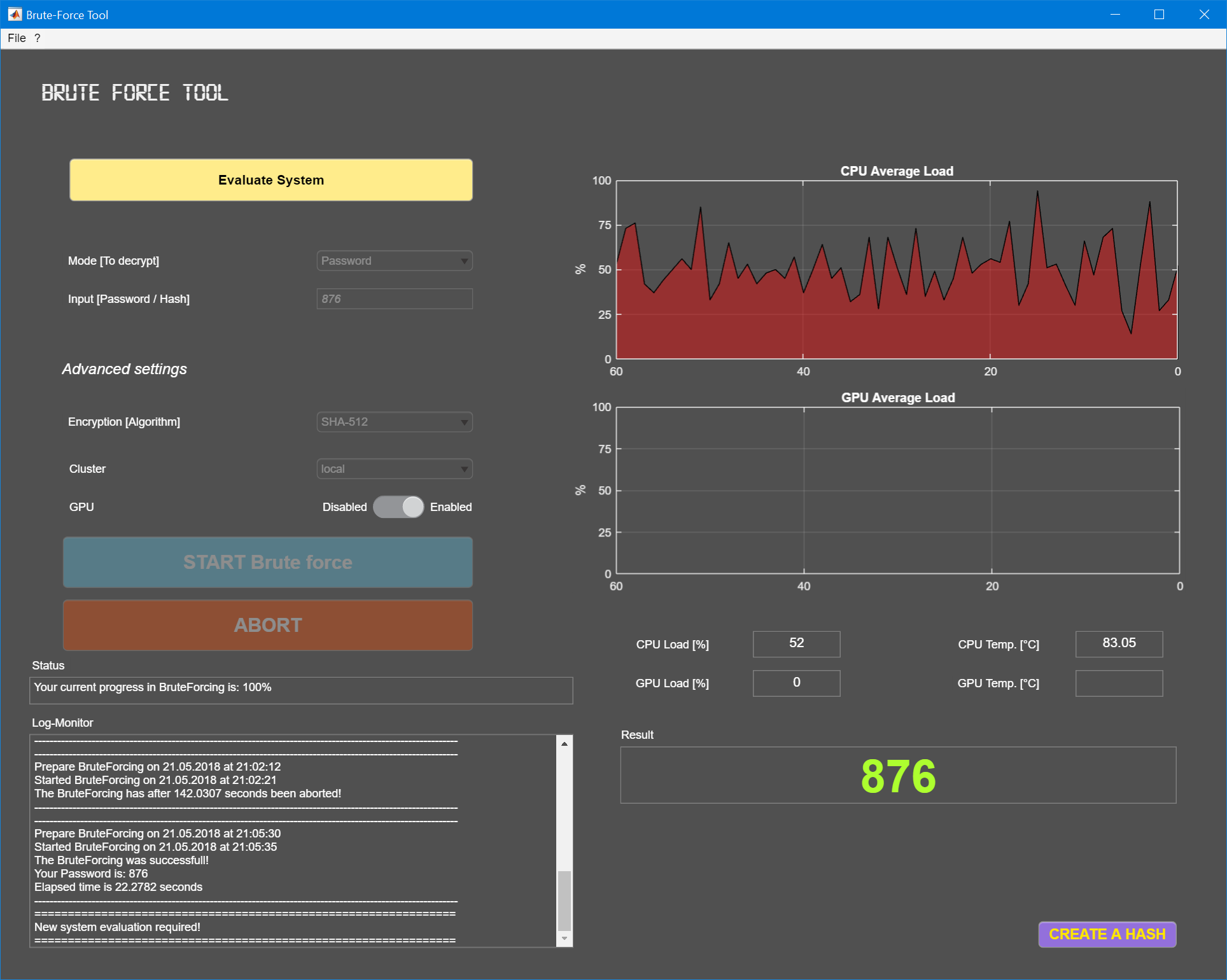


Abbildung Aktivieren der GPU

Sobald die GPU auf aktiv gesetzt wird und diese zuvor nie verwendet wurde, ist eine neue Systemevaluation notwendig. Dies wird dem User per Log-Monitor mitgeteilt. Des Weiteren wird das User Interface mit den GPU-Daten erweitert.

Sobald das System nochmals evaluiert wurde, kann die Software wie ab Kapitel 3.4 betrieben werden.

## Generieren eines Hashs

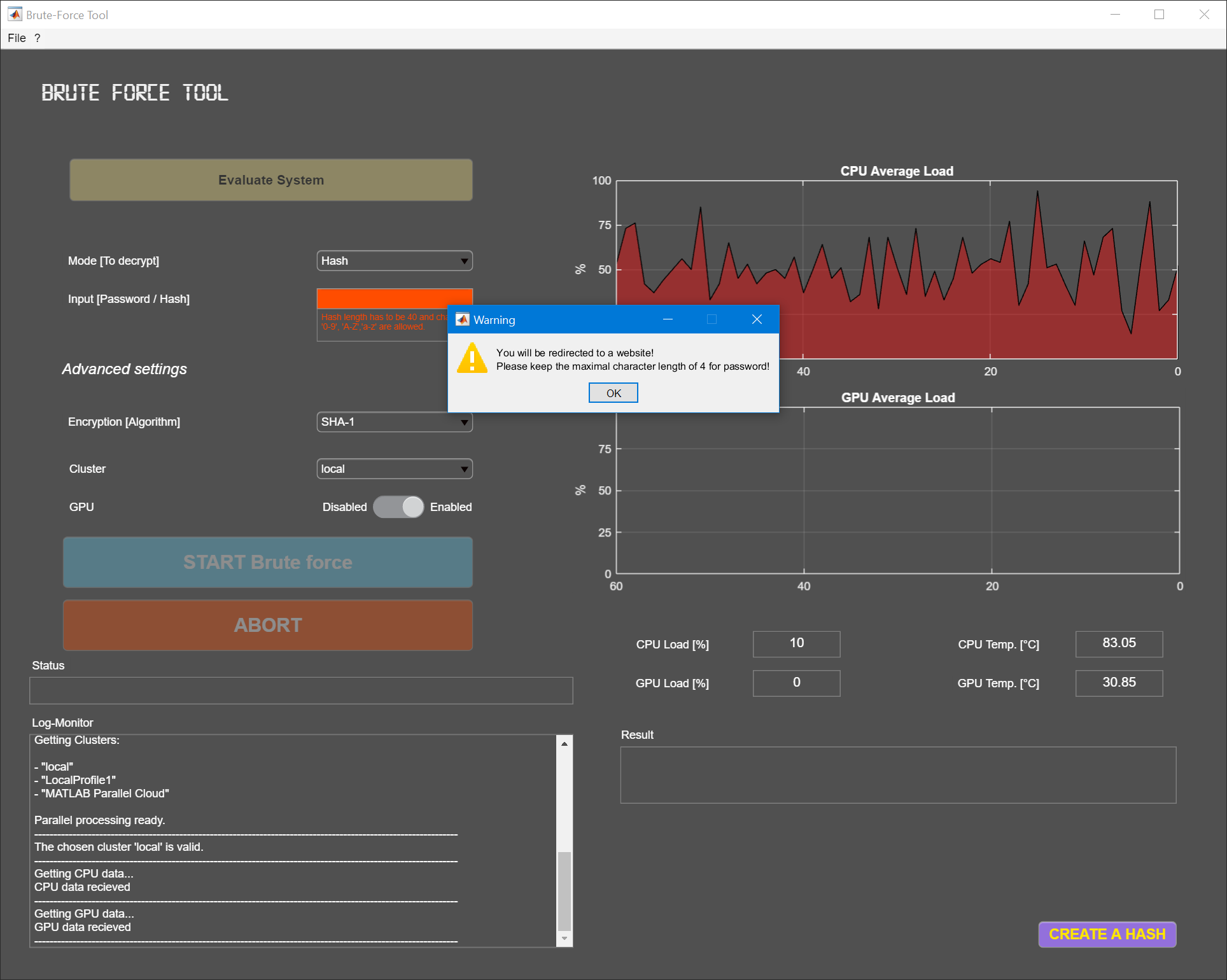


Abbildung Hash Generator

Sobald der Modus auf Hash gestellt wird, überprüft die Software das Eingabefeld. Nun muss dort zwingend ein Hash, welcher abhängig von der Verschlüsselung ist, eingetragen werden.

Dafür ist der Knopf «Create a hash» unten rechts vorgesehen. Wird dieser gedrückt erscheint eine Warnmeldung in der nochmals auf die Passwortlänge eingegangen wird. Durch bestätigen dieser Warnmeldung öffnet sich im Browserfenster der Hash Generator (siehe Abbildung 12).

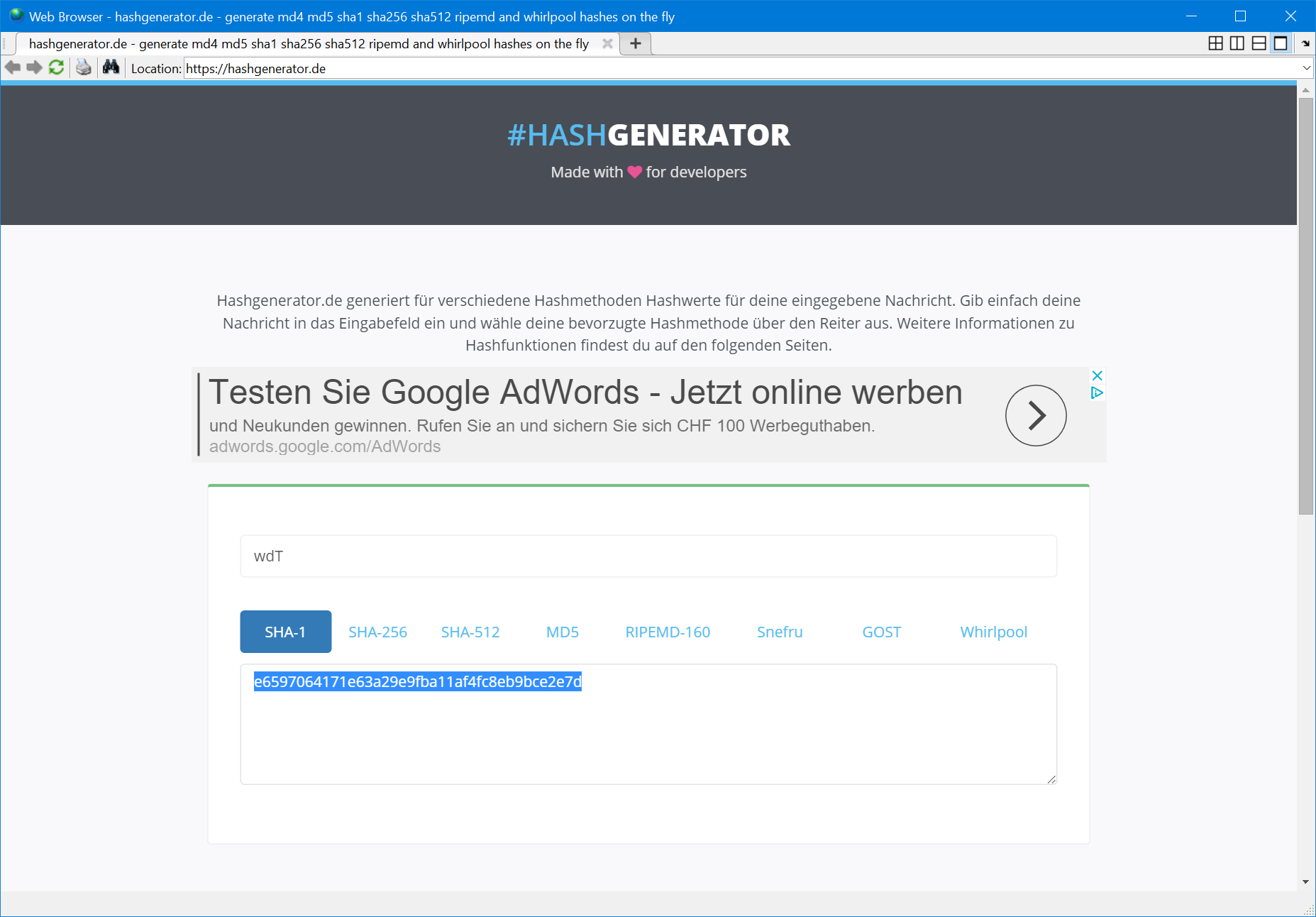
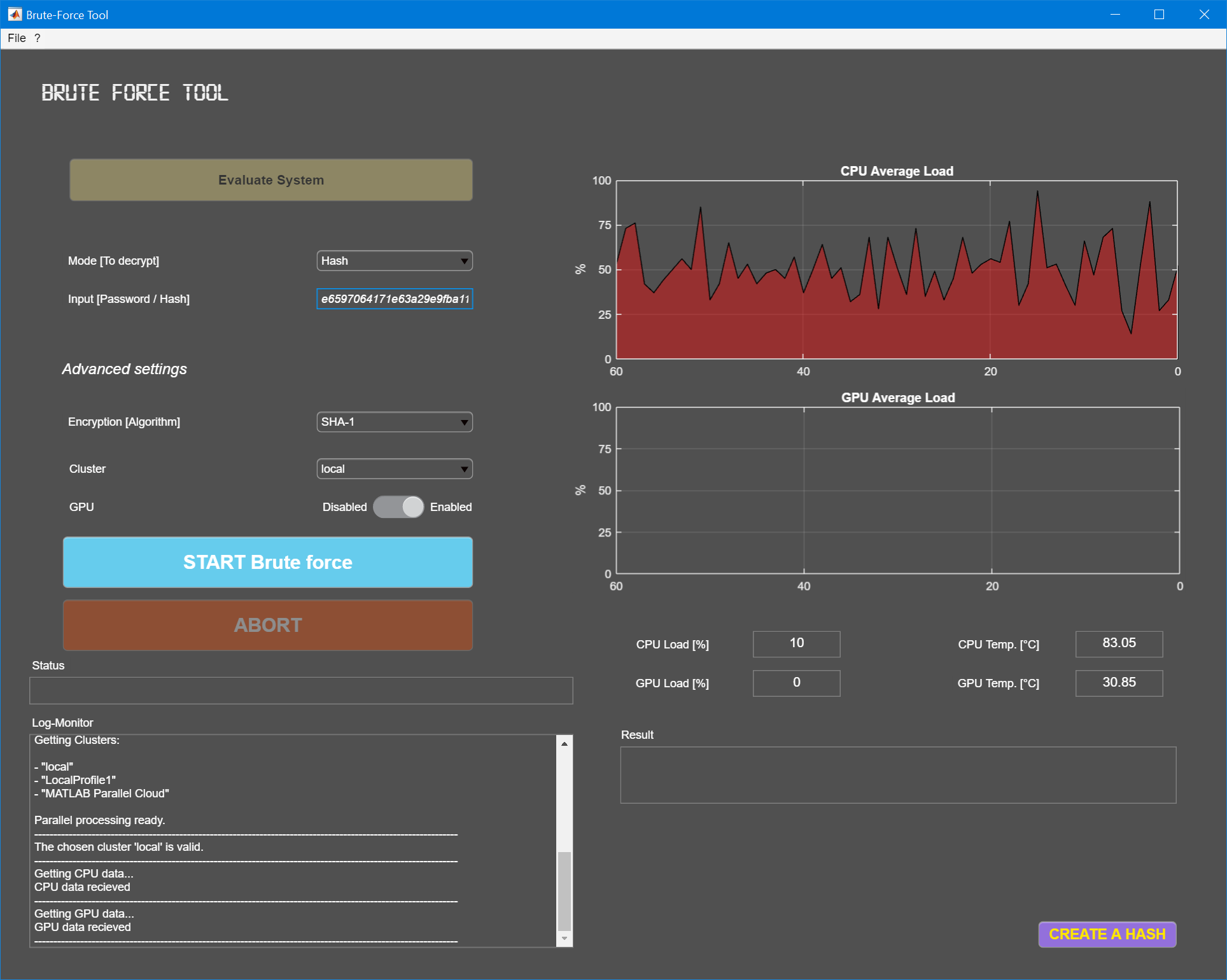
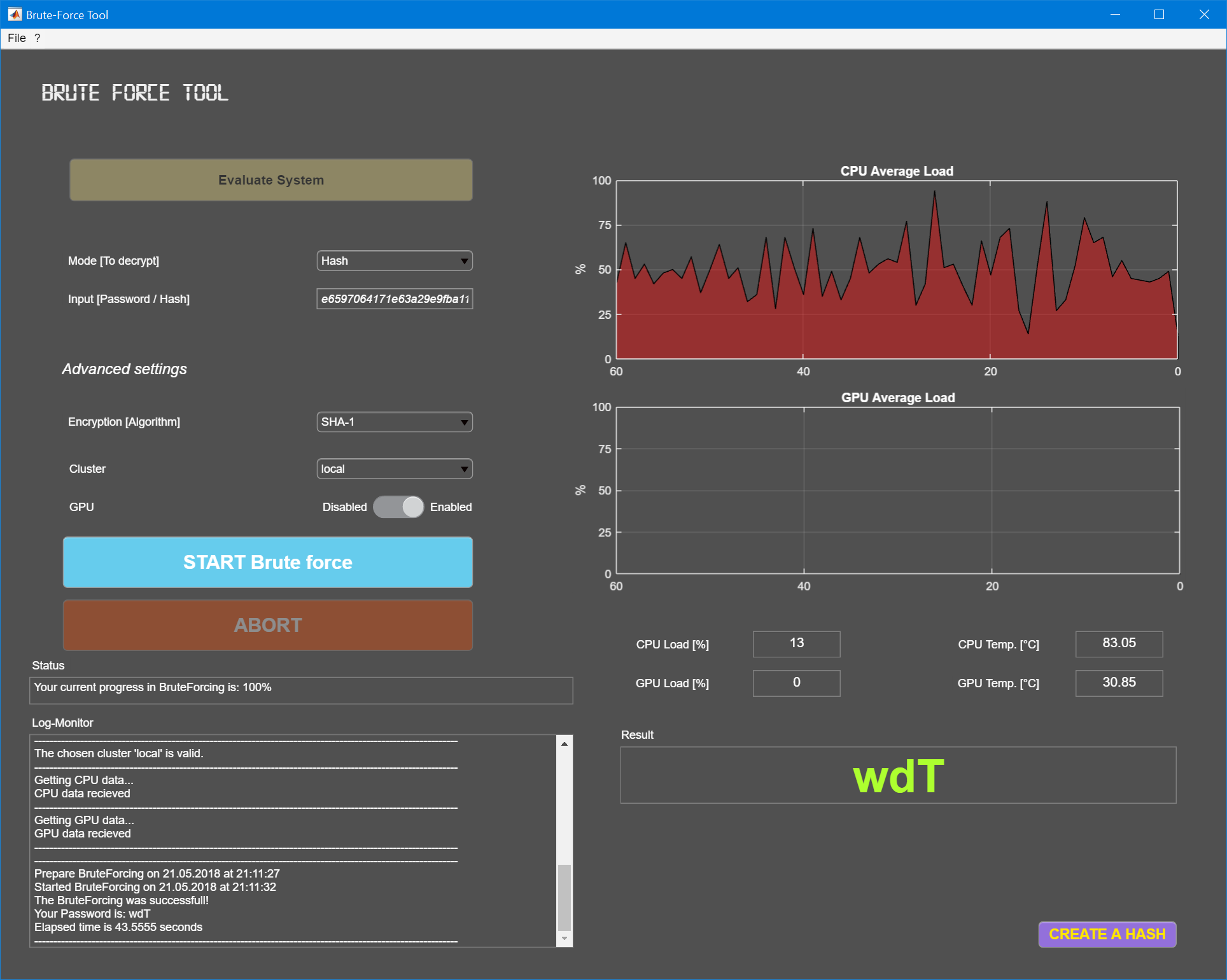


Abbildung Browserfenster mit dem Hash Generator





# Projektmanagement

## Soll – Ist Vergleich des Lastenheftes

## Commits

## Aufwand pro Funktion

|  |  |  |
| --- | --- | --- |
| **Funktion** | **Stunden (A. Gonzalez)** | **Stunden (B. Hürzeler)** |
| **BruteForce** |  |  |
| createString.m |  |  |
| DataHash.m |  |  |
| doBruteForce.m |  |  |
| doBruteForceAscendingly.m |  |  |
| doBruteForceAscendinglyUIUpdate.m |  |  |
| doBruteForceRandomly.m |  |  |
| initBruteForce.m |  |  |
| **Get-Set-Data** |  |  |
| Cpuinfo.m |  |  |
| displayData.m |  |  |
| getCpuData.m |  |  |
| getGpuData.m |  |  |
| saveFile.m |  |  |
| **UI** |  |  |
| deleteApp.m |  |  |
| initApp.m |  |  |
| userInterface\_script.m |  |  |
| **Scripts** |  |  |
| get-cpu-load.ps1 |  |  |
| get-gpu-load.ps1 |  |  |
| get-gpu-temperature.ps1 |  |  |
| get-temperature-TZ0-ps1 |  |  |
| **src** |  |  |
| runApp.m |  |  |
| systemFileBF.exe |  |  |
| **Dokumentation** |  |  |
|  |  |  |

Tabelle Zeitaufwand in Stunden

# Abnahmetests

## Abnahmetest der Module in «functions.cpp»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Lichtstärke messen | fgetLight | MiniQ auf ebenem Grund. | Aufruf Methode «fgetLight» durch «ebs2WDMiniQ» | Lichtstärke wird über analogen Eingang gemessen. | Wert liegt zwischen 0 und 5 V. | OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Winkel messen | fgetAngle | MiniQ auf ebenem Grund. | Aufruf Methode «fgetAngle» durch «ebs2WDMiniQ» | Winkel wird berechnet. | Wert liegt zwischen 0 und 360° | 120° -> Bedingt OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Kalibration Kompass | fcompassCalibrate | MiniQ auf ebenem Grund. | Aufruf Methode «fcompassCalibrate» durch «ebs2WDMiniQ» | Offsets x und y werden berechnet. | Offset x und Offset y werden im Speicher gesetzt. | OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| LED Farbe ändern | fsetColor | MiniQ auf ebenem Grund. | Setzen der verschiedenen Winkel und Aufruf Methode «fsetColor» durch «ebs2WDMiniQ» | LED wird zuerst grün dann gelb und danach rot leuchten. | LED leuchtet grün, gelb und rot | OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Taste einlesen | fgetKeyValue | MiniQ auf ebenem Grund. | Aufruf Methode «fgetKeyValue» durch «ebs2WDMiniQ» | Tasten 1,2 und 3 wird gedrückt und eingelesen. | Alle Tasten sollen erkannt werden. | OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Rechts drehen (Uhrezeigersinn) | fMoveProcedure | MiniQ auf ebenem Grund. | Aufruf Methode «fMoveProcedure» durch «ebs2WDMiniQ» | Winkel von Hand zwischen 0 und 180° setzen. | MiniQ dreht rechts herum.(Uhrzeigersinn) | OK |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Funktion** | **Methode** | **Ausgangslage** | **Ereignis** | **Vorgang** | **Ergebnis Soll** | **Ergebnis Ist** |
| Links drehen (Gegenuhrzeigersinn( | fMoveProcedure | MiniQ auf ebenem Grund. | Aufruf Methode «fMoveProcedure» durch «ebs2WDMiniQ» | Winkel von Hand zwi-schen 180 und 360° setzen. | MiniQ dreht links herum.(Gegenuhrzeigersinn) | OK |

## 

# Ehrlichkeitserklärung

«Hiermit erklären wir, die vorliegende Dokumentation selbständig, ohne Hilfe Dritter und nur unter Benutzung der angegebenen Quellen verfasst zu haben»

Herr Adrian Gonzalez, 24. Mai 2018, Windisch

Herr Bruno Hürzeler, 24. Mai 2018, Windisch