Server Logs konfigurieren und Debugging

Der Log Level und die Formatierung von BFFH Logzeilen können über folgende Wege konfiguriert werden.

Log Level per Umgebungsvariable

Über die Umgebungsvariable `BFFH_LOG=debug` - zum Beispiel eingebunden per systemd Service

Unterstützte Level für `BFFH_LOG` sind:

- info
- warn
- error
- debug
- trace

Log Parameter in bffhd

Über `bffhd` Parameter kann das Kommando beliebig angepasst werden (und ebenso im systemd Service verwendet werden):

`--log-format`

Gültige Werte sind:

- `Full` (Standard)
- `Compact`
- `Pretty`

> Die Groß- bzw. Kleinschreibung spielt hierbei keine Rolle!

Full

```
2024-12-07T11:10:48.198579Z DEBUG bffh:tls: difluoroborane::tls: TLS secret logging is disabled. keylog=false
2024-12-07T11:10:48.198685Z DEBUG bffh:tls: difluoroborane::tls: reading certificates
```

```
path=/etc/ssl/fablabchemnitz.de.cert.pem
2024-12-07T11:10:48.198959Z DEBUG bffh:tls: difluoroborane::tls: reading private key
path=/etc/ssl/fablabchemnitz.de.privkey.pem
2024-12-07T11:10:48.199608Z DEBUG difluoroborane::actors::process: Process actor updating
state name=actor-process-test cmd=/opt/fabinfra/actor-process-test.sh
state=SendState(ArchivedState { inner: ArchivedMachineState { state: Free, previous:
Some(ArchivedUserRef { id: "local_lab_admin" }) } })
2024-12-07T11:10:48.200451Z DEBUG difluoroborane::actors::process: Process actor updating
state name=Tasmota_Mjolnir cmd=/opt/fabinfra/adapters/tasmota/main.py
state=SendState(ArchivedState { inner: ArchivedMachineState { state: InUse(ArchivedUserRef {
id: "local_lab_admin" }), previous: Some(ArchivedUserRef { id: "local_lab_admin" }) } })
2024-12-07T11:10:48.203817Z  INFO bffh:binding API listen sockets: difluoroborane::capnp:
Opened listen socket on 127.0.0.1:5961
2024-12-07T11:10:48.204079Z  INFO bffh:binding API listen sockets: difluoroborane::capnp:
Opened listen socket on 192.168.1.192:5961
```

Compact

> Das kompakte Layout ist identisch zu Full

```
2024-12-07T11:10:48.198579Z DEBUG bffh:tls: difluoroborane::tls: TLS secret logging is
disabled. keylog=false
2024-12-07T11:10:48.198685Z DEBUG bffh:tls: difluoroborane::tls: reading certificates
path=/etc/ssl/fablabchemnitz.de.cert.pem
2024-12-07T11:10:48.198959Z DEBUG bffh:tls: difluoroborane::tls: reading private key
path=/etc/ssl/fablabchemnitz.de.privkey.pem
2024-12-07T11:10:48.199608Z DEBUG difluoroborane::actors::process: Process actor updating
state name=actor-process-test cmd=/opt/fabinfra/actor-process-test.sh
state=SendState(ArchivedState { inner: ArchivedMachineState { state: Free, previous:
Some(ArchivedUserRef { id: "local_lab_admin" }) } })
2024-12-07T11:10:48.200451Z DEBUG difluoroborane::actors::process: Process actor updating
state name=Tasmota_Mjolnir cmd=/opt/fabinfra/adapters/tasmota/main.py
state=SendState(ArchivedState { inner: ArchivedMachineState { state: InUse(ArchivedUserRef {
id: "local_lab_admin" }), previous: Some(ArchivedUserRef { id: "local_lab_admin" }) } })
2024-12-07T11:10:48.203817Z  INFO bffh:binding API listen sockets: difluoroborane::capnp:
Opened listen socket on 127.0.0.1:5961
2024-12-07T11:10:48.204079Z  INFO bffh:binding API listen sockets: difluoroborane::capnp:
Opened listen socket on 192.168.1.192:5961
```

Pretty

```
  2024-12-07T11:09:18.093419Z DEBUG difluoroborane::tls: reading certificates, path:
/etc/ssl/fablabchemnitz.de.cert.pem
    at bffhd/tls.rs:113
    in difluoroborane::tls::tls
    in bffh::bffh

  2024-12-07T11:09:18.093835Z DEBUG difluoroborane::tls: reading private key, path:
/etc/ssl/fablabchemnitz.de.privkey.pem
    at bffhd/tls.rs:123
    in difluoroborane::tls::tls
    in bffh::bffh

  2024-12-07T11:09:18.097839Z  INFO difluoroborane::capnp: Opened listen socket on
127.0.0.1:5961
    at bffhd/capnp/mod.rs:99
    in difluoroborane::capnp::binding API listen sockets
    in bffh::bffh

  2024-12-07T11:09:18.098140Z  INFO difluoroborane::capnp: Opened listen socket on
192.168.1.192:5961
    at bffhd/capnp/mod.rs:99
    in difluoroborane::capnp::binding API listen sockets
    in bffh::bffh
```

`--log-level`

Die Level sind die gleichen wie `BFFH_LOG`

Siehe auch Cheat Sheet - Wichtigste Befehle (Übersicht)

> Hinweis. Der `--log-level` Parameter funktioniert auf einer normalen Kommandozeilenausgabe derzeit nicht. Details finden sich in https://gitlab.com/fabinfra/fabaccess/bffh/-/issues/83. Als Workaround empfehlen wir die Verwendung der Umgebungsvariable `BFFH_LOG`.

`--verbose` (`-v`)

Dieser Parameter kann bis zu drei mal als Argument angegeben werden und erhöht die Log-Ausgabe zusätzlich. Beispiel:

```
/usr/bin/bffhd --config /etc/bffh/bffh.dhall --verbose --verbose --verbose
```

oder

```
/usr/bin/bffhd --config /etc/bffh/bffh.dhall -vvv
```

`--tls-key-log`

Dieser Parameter wird nur für Entwickler benötigt. Wenn für Debug Zwecke der Inhalt der verschlüsselten Verbindungen eingesehen werden soll, werden in der angegeben Datei `<PATH>` die Schlüssel für jede Verbindung gespeichert und können z.B. von Wireshark geladen werden.

Log File schreiben

Wer `bffhd` nicht über systemd startet und deshalb auch keine Logs mit journalctl auslesen möchte, der kann den Output auch konventiell in eine Log-Datei schreiben - hier im Beispiel `bffh.log`. Wir fangen dabei die Output-Streams `stdout` und `stderr` gemeinsam in einem Ausgabestrom ab (`2>&1`). Den Parameter `--log-level` verwenden wir nicht, da er bei der normalen Systemausgabe ignoriert wird (siehe Issue #83) und führen stattdessen mit `BFFH_LOG` an:

```
BFFH_LOG=debug /usr/bin/bffhd --config /etc/bffh/bffh.dhall --log-format Pretty > bffh.log
2>&1
```

Erweitertes Logging

Für Entwickler gibt es verschiedene Möglichkeiten aus der Laufzeit ein detaillierteres Fehlerbild zu erzeugen, über die Umgebungsvariable `RUST_BACKTRACE` zum Beispiel:

```
RUST_BACKTRACE=1 /usr/bin/bffhd
```

... oder ...

```
RUST_BACKTRACE=full /usr/bin/bffhd
```

Audit Log

Der Audit Log ist das Log File, was `bffhd` schreibt und in `bffh.dhall` konfiguriert wird. Es gibt Aufschluss über "Wer hat wann welche Ressource genutzt oder zurückgegeben?".

Details finden sich in Audit Log (Revisionsprotokoll). Es ist also kein Programmlog im eigentliche Sinne. Jedoch zeigt es auch die eventuell letzten Lebenszeichen des Servers an.

Debugging

Wer nicht nur Logs möchte, sondern debuggen will, der benötigt die Laufzeit inklusive Debug Symbolen. Im Falle der Installation mit *.deb oder *.rpm Paketen gibt es dafür gesonderte Pakete. Alternativ kann BFFH auch mit `cargo` kompiliert werden. Letzteres erlaubt das Beeinflussen des Builds mit entsprechenden Parametern. Siehe BFFH testen / entwickeln.

Hilfreiche Debugging Tools

- `strace` - ein in Linux-Distros standardmäßig eingebautes Kommando
- `gdb` - GNU Debugger
- `valgrind` - https://valgrind.org

Versionsinformationen anzeigen

Welche BFFH Version läuft, bzw. ist installiert? Un mit welchem Branch, Commit ID und Rustumgebung wurde `bffhd` gebaut? Siehe hier: Version anzeigen

Bugs melden

Bugs können zum einen in den Community Kanälen besprochen werden und/oder auf GitLab gemeldet werden.

Version #27
Erstellt: 23 Oktober 2024 22:53:07 von Mario Voigt (Stadtfabrikanten e.V.)
Zuletzt aktualisiert: 17 März 2025 16:54:07 von Mario Voigt (Stadtfabrikanten e.V.)