

# Sonstige Software

- Stempel aus Adobe Acrobat exportieren
- Solarwinds Dameware
  - Standard-Anmeldeinformationen für Remote Verbindungen einstellen
- Wallpaper Engine
  - Hintergrund lässt sich nicht ändern
- Heimdall
  - Schaltflächen mit "Blur-Effekt" belegen
- Invoice Ninja
  - Invoiceninja zeigt nicht alle Währungen oder Sprachen an
- OpenSSL
  - Neue Zertifikate mit OpenSSL erstellen
  - PFX Zertifikat in .crt und .key Datei exportieren
  - PEM-Zertifikat in CRT-Zertifikat konvertieren
- Wireguard
  - MTU Größe für Wireguard Verbindung berechnen
  - WireGuard mit WebInterface auf Debian Server installieren

# Stempel aus Adobe Acrobat exportieren

## Einleitung

Wenn du die Stempel im Acrobat Reader exportieren möchtest, musst du die Stempel als Dateien sichern. Diese Dateien kannst du dann in anderen Installationen wieder integrieren.

## Speicherort Stempel

Um die Stempel zu sichern, öffnest du den **Datei Explorer**. Dort navigierst du zu folgendem Pfad

```
C:\Users\<benutzer>\AppData\Roaming\Adobe\<version>\Stamps
```

Dort befinden sich die PDF-Dateien. Die Stempel werden als PDF Datei abgespeichert. Diese kannst du dir dann abspeichern oder in eine andere Installation wieder integrieren.

# Solarwinds Dameware

# Standard-Anmeldeinformationen für Remote Verbindungen einstellen

## Einleitung

In diesem Artikel beschreibe ich kurz, wie wir in **Dameware Version 12** einstellen können, welche **Standard-Anmeldeinformationen** der **Remote-Client** beim Verbinden, verwenden soll.

Dadurch müssen wir nicht bei jeder Verbindung die z.B. **Administrator Anmeldedaten** angeben.

## Anmeldeinformationen einstellen

Im ersten Schritt müssen wir den **Mini Remote Control Client** öffnen. Dieser sollte zwischen den installierten Programmen auftauchen. Dort navigieren wir zu den Einstellungen über "**Ansicht**" => "**Standard Hosteinstellungen**".

Dort geben wir unter "**Authentifizierungsoptionen**" die **Anmeldeinformationen** ein. Dazu müssen wir noch den Haken bei "**Sicherheits-Anmeldeinformationen merken**" setzen. Wenn wir jetzt auf **Übernehmen** klicken, haben wir die Einstellungen gesetzt.

# Wallpaper Engine

# Hintergrund lässt sich nicht ändern

## Einleitung

In diesem Artikel geht es kurz darum, wie wir das oben genannte Problem mit der *Wallpaper Engine (Steam Edition)* beseitigen können. Ich bin darauf gestoßen, da ich mein Hintergrundbild leider nicht ändern konnte. Selbst ein Neustart der Anwendung hatte nichts gebracht.

## Problemlösung

Die Lösung des Problems ist ziemlich simpel. Es muss lediglich die Einstellungsdatei des Programms gelöscht werden. Diese finden wir unter dem folgenden Pfad:

```
C:\Program Files (x86)\Steam\steamapps\common\wallpaper_engine\config.json
```

Wenn wir die Datei gelöscht haben, müssen wir einmal die Anwendung neu starten. Sobald wir dies gemacht haben, erhalten wir wieder ein "Willkommen" Fenster und müssen die Software einmal wieder neu einstellen. Danach sollten wir aber in der Lage sein, das Hintergrundbild zu ändern.

# Heimdall

# Schaltflächen mit "Blur-Effekt" belegen

## Einleitung

Beim Erstellen eines Dashboards mit **Heimdall** wollte ich gerne die Schaltflächen mit einem "**Blur Effekt**" belegen. Dafür müssen wir nur den nachstehenden **CSS-Code** in das System einpflegen. Dazu einfach die Einstellungen öffnen und den Code in das Textfeld eingeben.

## Code

```
@import url("https://fonts.googleapis.com/css2?family=Raleway:wght@700;800&display=swap");

#app {
  background-position: right;
}
#app #sortable,
#app main {
  padding: 20px;
}
#config-buttons {
  bottom: 50%;
  transform: translateY(50%);
  border-top-left-radius: 10px;
  border-bottom-left-radius: 10px;
  box-shadow: rgba(255, 255, 255, 0.1) -1px 1px 1px 0, rgba(255, 255, 255, 0.1) 0 -1px 1px 0,
    rgba(0, 0, 0, 0.1) -1px 0 20px 5px;
  background-color: rgba(40, 40, 40, 0.25);
  backdrop-filter: blur(10px);
  -webkit-backdrop-filter: blur(10px);
}
#config-buttons a {
  background: none;
}
```



```
#config-buttons a svg {
  transition: all 0.1s ease-in-out;
  color: rgba(255, 255, 255, 0.5);
}

#config-buttons a:hover svg {
  transform: scale(1.1);
  color: rgba(255, 255, 255, 0.95);
}

.black {
  color: white !important;
}

.item {
  box-shadow: rgba(0, 0, 0, 0.05) -1px -1px 5px 0, rgba(0, 0, 0, 0.15) 0px 20px 25px -5px,
    rgba(0, 0, 0, 0.04) 0px 10px 10px -5px !important;
  border-radius: 12px;
  background-image: none;
  border: none;
  outline: none;
  height: 100px;
  width: 300px;
  margin: 1.25rem;
  padding: 1rem 55px 1rem 1rem;
  transition: all 0.25s ease-in-out;
  transition-property: transform, box-shadow, background-color;
  background-color: rgba(255, 255, 255, 0.4) !important;
  backdrop-filter: blur(10px);
  -webkit-backdrop-filter: blur(10px);
}

.item:after {
  height: 100px;
  opacity: 0.2;
}

.item:hover {
  transform: scale(1.1);
  background-color: rgba(255, 255, 255, 0.2) !important;
  box-shadow: rgba(0, 0, 0, 0.1) 0px 60px 40px -7px !important;
}

.item .svg-inline--fa {
```

```
height: 100px;
vertical-align: middle;
opacity: 0.2;
}
```

```
@media only screen and (max-width: 750px) {
  .item {
    padding-top: 2rem;
    padding-bottom: 2rem;
  }
  #config-buttons {
    display: none;
  }
}
```

```
.details * {
  color: white !important;
}
.details {
  padding: 0 0.5rem;
}
```

```
.app-icon {
  filter: drop-shadow(1px 1px 2px rgba(10, 0, 20, 0.1));
}
```

```
.livestats-container {
  margin-top: 0.5rem;
}
.livestats-container .livestats .title {
  margin-bottom: 3px;
}
.livestats-container .livestats li {
  padding-right: 1rem;
}
```

```
.livestats-container strong {
  font-weight: 500 !important;
  padding: 0 2px;
}
```

```
.details > .title {  
  font-weight: 800;  
  font-size: 1.3rem !important;  
  letter-spacing: 1px;  
  font-family: "Raleway", sans-serif;  
  text-shadow: rgba(10, 0, 60, 0.25) 1px 0 5px;  
  text-shadow: -1px -1px 1px rgba(255, 255, 255, 0.1), 1px 1px 2px rgba(10, 0, 60, 0.25);  
  transition: all 0.25s ease-in-out;  
}  
.item:hover .details > .title {  
  text-shadow: -1px -1px 1px rgba(255, 255, 255, 0.1), 1px 1px 5px rgba(10, 0, 60, 0.2);  
}  
  
.item-container .tooltip {z-index: -1; }
```

**Quelle:**

[https://www.reddit.com/r/selfhosted/comments/nzw76z/i\\_went\\_a\\_bit\\_overboard\\_customising\\_heimdall/](https://www.reddit.com/r/selfhosted/comments/nzw76z/i_went_a_bit_overboard_customising_heimdall/)

# Invoice Ninja

# Invoiceninja zeigt nicht alle Währungen oder Sprachen an

## Einleitung

Invoice Ninja ist ein kleines Tool, mit dem man ganz schnell PDF Rechnungen erstellen kann, um diese unter anderem an Kunden zu schicken. So muss man keine Excel Listen mehr pflegen. Das Programm gibt einem eine Übersicht der offenen Salden der Kunden als auch eine gesamte Übersicht der Kunden.

Dieses Programm kann bei der Installation ein Problem verursachen, womit es nicht möglich ist, die Software in der deutschen Sprache als auch Rechnungen mit Euro auszuweisen. Um dieses Problem zu lösen, müssen wir einen Befehl in der URL Seite der Internetseite auslösen.

## Anwendung

Zuerst melden wir uns bei unserer Invoice Ninja Instanz an und gehen ganz normal auf die Startseite. In der URL Zeile schreiben wir folgenden Text hinten ran:

```
/update?secret=secret
```

Der Link sollte dann folgendermaßen aussehen:

```
https://invoiceninja.domain.de/update?secret=secret
```

Jetzt sollte er die Datenbank neu angefragt haben und die entsprechenden Einträge sollten jetzt verfügbar sein.

# OpenSSL

# Neue Zertifikate mit OpenSSL erstellen

## Einleitung

In dieser Anleitung werden wir mithilfe von **OpenSSL TLS Zertifikate** erstellen, mit dem wir dann unseren **Web-Server** über **HTTPS** erreichbar machen können. **OpenSSL** ist für diesen Zweck eine passende Alternative, da diese keine Verbindung ins Internet benötigt, und **kostenlos** verwendbar ist.

Um diese Zertifikate zu erstellen, müssen wir sicherstellen, dass wir das Paket `openssl` installiert haben. Wenn dies nicht der Fall ist, können wir dieses eben nachinstallieren.

```
apt update && apt install openssl -y
```

## Erstellen von PK und CSR

In diesem Abschnitt werden wir einen **PK (Private Key)** und einen **CSR (Certificate Signing Requests)** erstellen. Mit diesen Dateien können wir dann unseren **Web-Server verschlüsseln**.

```
openssl req -newkey rsa:2048 -nodes -keyout domain.key -out domain.csr
```

Wenn wir diesen Befehl abgesetzt haben, müssen wir die Abfragen, die Folgen beantworten. Der erste Parameter `-newkey rsa:2048` gibt an, dass der Schlüssel **2048 bit** lang sein soll, und mit dem **RSA Algorithmus** verschlüsselt werden soll.

Mit dem zweiten Parameter `-nodes` geben wir an, dass wir den **Schlüssel nicht** mit einem **Kennwort verschlüsseln**.

## CSR aus einem vorhandenen PK erstellen

Der folgende Befehl kann verwendet werden, wenn wir ein **CSR (Certificate Signing Requests)** erstellen möchten, wir aber schon einen **Private Key** besitzen, bzw. erstellt haben.

```
openssl req -key domain.key -new -out domain.csr
```

Jetzt müssen wir beim ersten Parameter `-key` den Dateinamen unseres Schlüssels angeben. Sobald wir das eingetragen haben, können wir den Befehl abschicken und die **CSR Datei** wird erstellt.

## Selbst signiertes SIL-Zertifikat erstellen

Jetzt wollen wir ein **TLS Zertifikat** erstellen, welches nicht von einer externen Zertifizierungsstelle stammt. Dies können wir uns von Gebrauch machen, wenn wir unseren Web-Server verschlüsseln möchten.

```
openssl req -newkey rsa:2048 -nodes -keyout domain.key -x509 -days 365 -out domain.crt
```

Der erste Parameter gibt wieder den **Verschlüsselungsalgorithmus** an. Der Parameter `-x509` gibt an, dass wir unser Zertifikat nicht von einer externen Zertifikatsstelle signieren lassen möchten. Und der folgende Parameter `-days` gibt an, wie lange unser Zertifikat gültig ist.



# PFX Zertifikat in .crt und .key Datei exportieren

## Einleitung

Um Zertifikate in bestimmte Software einspielen zu können, benötigen wir manchmal die **.crt** und die **.key** Datei. Dazu können wir uns das **OpenSSL-Tool** zu Hilfe nehmen.

## Zertifikat exportieren

### .crt Datei exportieren

Um die **.crt Datei** zu erhalten, müssen wir nur den folgenden Befehl absetzen. Dabei müssen wir im ersten Schritt den Datei-Namen der **.pfx-Datei** anpassen. Beim Absenden des Befehls geben wir nur noch das Kennwort ein, mit dem das Zertifikat verschlüsselt wurde.

```
openssl pkcs12 -in <PFX-Zertifikat> -clcerts -nokeys -out <CRT-Datei>
```

#Beispiel:

```
openssl pkcs12 -in cert.pfx -clcerts -nokeys -out cert.crt
```

### .key Datei exportieren

Um jetzt die **.key Datei** zu erhalten, müssen wir diesmal 2 Befehle absetzen. Einmal um die **.key-Datei** zu erhalten, und im zweiten Schritt um die **.key-Datei** zu entschlüsseln.

```
openssl pkcs12 -in cert.pfx -nocerts -out cert-encrypted.key
```

Im Anschluss folgt jetzt die **Entschlüsselung** der **.key-Datei**.

```
openssl rsa -in cert-encrypted.key -out cert.key
```

Damit haben wir dann beide Dateien erfolgreich exportiert, um diese dann in bestimmten Anwendungen zu verwenden.



# PEM-Zertifikat in CRT-Zertifikat konvertieren

In diesem Beitrag geht es kurz darum, wie wir ein **PEM-Zertifikat** in ein **CRT-Zertifikat** konvertieren können. Benötigt wird dafür eine Installation von **OpenSSL**.

```
openssl x509 -outform der -in "C:\<Pfad>\cert.pem" -out "C:\<Pfad>\cert.crt"
```

# Wireguard

# MTU Größe für Wireguard Verbindung berechnen

## Einleitung

Bei der Einrichtung von einem **Wireguard Server** sollte eine **MTU-Size** mitgegeben werden. Diese gibt an, wie groß ein **VPN-Paket** mit Nutzdaten voll sein darf. In dieser Anleitung berechnen wir kurz unsere **MTU-Size**, um so die höchste **Performance** aus unserem **Wireguard Server** herauszuholen.

Wireguard verwendet eine **Standard MTU** von **1420**. Dies kann zu Problemen führen wenn dadurch das **Paket** zu groß ist und somit nicht an den **Server** oder an den **Client** übermittelt werden kann.

## Berechnung

Die Berechnung unterscheidet sich je nachdem welchen Anschluss ihr als eure Internetanbindung verwendet. Wenn es im Einzelfall nicht funktionieren sollte, prüft bitte, welche MTU-Größe ihr für euren Provider benötigt. Ich berechne die MTU-Größe beispielsweise für **DSL** mit einer **MTU** von **1492** und für **Kabel-Internet** mit einer **MTU** von **1500**.

Anhand der nachfolgenden Tabelle können die einzelnen **Byte-Größen** entnommen werden, die zur Berechnung der **MTU** benötigt werden:

Beschreibung	Byte Größe
IPv4-Verbindung	20 Bytes
IPv6-Verbindung	40 Bytes
UDP-Paket	8 Bytes
Wireguard Overhead	32 Bytes

Dementsprechend lassen sich folgende Beispiele berechnen:

Beispiel 1	Beispiel 2
MTU DSL	MTU Kabel-Internet

1492 MTU (Provider)	1500 MTU (Provider)
- 20 (IPv4)	- 40 (IPv6)
- 8 (UDP)	- 8 (UDP)
- 32 (Wireguard Overhead)	- 32 (Wireguard Overhead)
<b>= MTU 1432</b>	<b>= MTU 1420</b>

Die **DSL Wireguard Verbindung** verwendet im Rechenbeispiel eine Verbindung über **IPv4**, wohin gegen die **Kabel Wireguard Verbindung** eine Verbindung über **IPv6** herstellen möchte.

# WireGuard mit WebInterface auf Debian Server installieren

## Einleitung

Wenn wir unseren eigenen **WireGuard Server** betreiben möchten, wollen wir diesen vielleicht ja auch über eine **Weboberfläche administrieren**. Als Oberfläche bedienen wir uns dann an **wireguard-ui** [Link](#).

Mithilfe von **WireGuard** können wir dann ganz einfach eine **VPN-Verbindung** in unser Netzwerk herstellen und so einerseits auf unser **Heimnetzwerk** zugreifen, als auch **verschlüsselten Datenverkehr** nutzen.

## Vorbereitungen durchführen und WireGuard installieren

Im ersten Schritt installieren wir alle benötigten Pakete auf unserem **Debian Server**, damit wir **WireGuard** betreiben können.

```
apt update && apt install -y wireguard curl iptables tar  
cd /etc/wireguard
```

Wir müssen auch sicherstellen das unsere **Firewall Port 51820/udp** durchlässt. Und das wir ggf. unseren **WireGuard Server** als **Exposed Host** oder durch **Port NAT** unseren Server von außen erreichbar machen. Wie genau das funktioniert, bitte dem Handbuch des Herstellers der Firewall entnehmen.

Jetzt müssen wir das **IP-Forwarding** aktivieren. Dies hat den Hintergrund, dass der gesamte Verkehr auf dem **WireGuard Interface** weitergeleitet wird.

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf  
echo "net.ipv6.conf.all.forwarding=1" >> /etc/sysctl.conf  
sysctl -p
```

# WireGuard UI installieren

Jetzt müssen wir folgenden Code ausführen, damit wir das **Startskript** für **WireGuard UI** erstellt wird. Damit wird der **Webserver** gestartet, und die **Datenbank** unter `/etc/wireguard/db/` abgelegt.

```
cat <<EOF > /etc/wireguard/start-wgui.sh
#!/bin/bash

cd /etc/wireguard
./wireguard-ui -bind-address 0.0.0.0:5000
EOF
chmod +x start-wgui.sh
```

Mit dem folgenden Skript wird **WireGuard UI** installiert und die entsprechenden **Dienste** angelegt. Im Anschluss wird der Download von **WireGuard UI** gestartet und der **Web Server** wird gestartet.

```
cat <<EOF > /etc/systemd/system/wgui-web.service
[Unit]
Description=WireGuard UI

[Service]
Type=simple
ExecStart=/etc/wireguard/start-wgui.sh

[Install]
WantedBy=multi-user.target
EOF

cat <<EOF > /etc/wireguard/update.sh
#!/bin/bash

VER=$(curl -sI https://github.com/ngoduykhanh/wireguard-ui/releases/latest | grep "location:" | cut -d "/" -f8 | tr -d '\r')

echo "downloading wireguard-ui \${VER}"
curl -sL "https://github.com/ngoduykhanh/wireguard-ui/releases/download/\${VER}/wireguard-ui-\${VER}-linux-amd64.tar.gz" -o wireguard-ui-\${VER}-linux-amd64.tar.gz

echo -n "extracting "; tar xvf wireguard-ui-\${VER}-linux-amd64.tar.gz
```



```
echo "restarting wgui-web.service"  
systemctl restart wgui-web.service  
EOF
```

```
chmod +x /etc/wireguard/update.sh  
cd /etc/wireguard; ./update.sh
```

## Änderungen der Konfiguration verarbeiten

Mit dem folgenden Skript wird die **WireGuard Schnittstelle wg0** beobachtet. Bei Veränderungen an der Konfigurationsdatei wird dann **WireGuard** neu gestartet.

```
cat <<EOF > /etc/systemd/system/wgui.service  
[Unit]  
Description=Restart WireGuard  
After=network.target  
  
[Service]  
Type=oneshot  
ExecStart=/bin/systemctl restart wg-quick@wg0.service  
  
[Install]  
RequiredBy=wgui.path  
EOF
```

```
cat <<EOF > /etc/systemd/system/wgui.path  
[Unit]  
Description=Watch /etc/wireguard/wg0.conf for changes  
  
[Path]  
PathModified=/etc/wireguard/wg0.conf  
  
[Install]  
WantedBy=multi-user.target  
EOF
```

## WireGuard und WireGuard UI starten

Jetzt starten wir **WireGuard** und **WireGuard UI**.

```
touch /etc/wireguard/wg0.conf  
systemctl enable wgui.{path,service} wg-quick@wg0.service wgui-web.service  
systemctl start wgui.{path,service}
```

Wir können jetzt das **Web Interface** von **WireGuard UI** unter der **Public IP** und **Port 5000** öffnen. Die Installation ist damit abgeschlossen!

wireguard wireguardui installation.png