WolfISMS

Wolf's Informations-Sicherheits-Management-System

Docker-Installation

Version 3.63

Stand: August 2026

1. Überblick

Diese Anleitung beschreibt den Betrieb von WolfISMS als Docker-Container hinter dem Nginx Proxy Manager (NPM). NPM übernimmt dabei die TLS-Terminierung und die Zertifikatsverwaltung, WolfISMS selbst spricht ausschließlich HTTP innerhalb des Docker-Netzwerks.

Sie richtet sich an Administratoren, die eine bestehende Docker-Umgebung betreiben. Für die klassische Installation direkt auf einem Ubuntu- oder Debian-Server mit Apache und mod_wsgi siehe stattdessen Kapitel 3 der Administrationsdokumentation.

1.1 Zielarchitektur

Komponente	Aufgabe
Nginx Proxy Manager	Nimmt HTTP (80) und HTTPS (443) von außen entgegen, terminiert TLS, verwaltet die Let's-Encrypt-Zertifikate und leitet die Anfragen im internen Netz an den WolfISMS-Container weiter. Die Administrationsoberfläche läuft auf Port 81
WolfISMS-Container	Die Anwendung selbst. Sie ist von außen nicht direkt erreichbar, sondern ausschließlich über das Docker-Netzwerk proxy
Datenbank	MariaDB bzw. MySQL – je nach Betriebsmodell als eigener Container oder als bereits vorhandener Datenbankserver
Docker-Netzwerk "proxy"	Verbindet NPM und WolfISMS. Es wird einmalig manuell angelegt und von beiden Compose-Projekten als externes Netz eingebunden

1.2 Warum HTTPS zwingend erforderlich ist

WolfISMS setzt seine Session-Cookies mit dem Attribut Secure=True. Ein Browser überträgt solche Cookies ausschließlich über verschlüsselte Verbindungen. Über reines HTTP kommt daher zwar die Loginseite zustande, die Anmeldung schlägt aber fehl, weil die Session nicht gehalten werden kann. Ein gültiges Zertifikat ist damit keine Empfehlung, sondern Betriebsvoraussetzung.

Reverse-Proxy-Unterstützung ist eingebaut:
Die Anwendung wertet die Header X-Forwarded-Proto und X-Forwarded-Host aus (ProxyFix in isms.py). Sie erkennt dadurch korrekt, dass sie hinter einem TLS-Terminierungspunkt läuft, und erzeugt absolute Links mit https und dem öffentlichen Hostnamen. Der Nginx Proxy Manager setzt diese Header standardmäßig – eine zusätzliche Konfiguration ist nicht nötig.

2. Voraussetzungen

Komponente	Anforderung
Docker Engine	Aktuelle Version mit dem Plug-in Compose V2 (docker compose, nicht docker-compose)
Betriebssystem	Beliebiger Docker-Host (Linux empfohlen)
Öffentlicher DNS-Eintrag	Der gewünschte Hostname, z. B. isms.ihre-domain.de, muss auf die öffentliche IP-Adresse des Docker-Hosts zeigen
Portfreigaben	Die Ports 80 und 443 müssen von außen auf den Docker-Host erreichbar sein. Port 80 wird für die ACME-Challenge von Let's Encrypt benötigt
Port 81	Die NPM-Administrationsoberfläche. Sie sollte nicht öffentlich erreichbar sein, sondern nur aus dem internen Netz oder über VPN
Installationspaket	Das WolfISMS-Repository bzw. Docker-Paket mit docker-compose.yml und .envdefault (vom Hersteller bereitgestellt)
Datenbank	MariaDB 10.11+ oder MySQL 8.0+ – als Container oder als externer Server

Port 81 nicht ins Internet veröffentlichen:
Die Administrationsoberfläche des Nginx Proxy Managers verwaltet sämtliche Zertifikate und Weiterleitungen. Beschränken Sie den Zugriff über die Firewall des Hosts auf das interne Netz und ändern Sie die Standard-Zugangsdaten unmittelbar nach der ersten Anmeldung.

3. Schritt 1 – Docker-Netzwerk anlegen

Beide Compose-Projekte – der Nginx Proxy Manager und WolfISMS – binden dasselbe externe Netzwerk ein. Es wird auf dem Docker-Host einmalig angelegt:

```
docker network create proxy
```

Die Prüfung, ob das Netz existiert:

```
docker network ls | grep proxy
```

Das Netzwerk muss vor dem ersten docker compose up bestehen. Fehlt es, brechen beide Projekte mit der Meldung „network proxy declared as external, but could not be found“ ab.

4. Schritt 2 – Nginx Proxy Manager bereitstellen

Legen Sie auf dem Docker-Host ein eigenes Verzeichnis für den Reverse Proxy an und darin die Datei docker-compose.yml:

```
mkdir -p /opt/npm
cd /opt/npm
```

Inhalt der Datei /opt/npm/docker-compose.yml:

```
services:
  npm:
    image: jc21/nginx-proxy-manager:2
    restart: unless-stopped
    ports:
      - "80:80"     # HTTP + ACME-Challenge
      - "443:443"   # HTTPS
      - "81:81"     # Administrationsoberflaeche
    volumes:
      - npm_data:/data
      - npm_letsencrypt:/etc/letsencrypt
    networks:
      - proxy

networks:
  proxy:
    external: true
    name: proxy

volumes:
  npm_data:
  npm_letsencrypt:
```

Anschließend den Container starten:

```
docker compose up -d
```

4.1 Erste Anmeldung

Die Administrationsoberfläche ist danach unter http://<docker-host>:81 erreichbar. Der Nginx Proxy Manager legt beim ersten Start ein Standardkonto an, dessen Zugangsdaten in der Dokumentation des Images genannt werden.

> Ändern Sie unmittelbar nach der ersten Anmeldung E-Mail-Adresse und Passwort des Standardkontos. Solange die Vorgabewerte gelten, kann jeder mit Netzzugriff auf Port 81 die Zertifikate und Weiterleitungen der gesamten Umgebung verändern.

5. Schritt 3 – WolfISMS bereitstellen

Das WolfISMS-Paket enthält die docker-compose.yml sowie die Vorlage .envdefault. Die eigentliche Konfiguration erfolgt ausschließlich über Umgebungsvariablen in der Datei .env.

5.1 Paket beziehen und Konfiguration anlegen

```
git clone https://github.com/wolfschus/wolfisms.git
cd WolfISMS
cp .envdefault .env
```

Danach die Datei .env öffnen und die Werte an die eigene Umgebung anpassen. Die vollständige Referenz aller Variablen enthält Kapitel 8.

Werden eigene Zertifikate benötigt – etwa für die Verbindung zu einem externen Datenbankserver –, werden diese in das Verzeichnis certs/ kopiert. Für die TLS-Terminierung nach außen ist das nicht erforderlich; darum kümmert sich der Nginx Proxy Manager.

5.2 Container starten

```
docker compose up -d
```

Den Start kontrollieren:

```
docker compose ps
docker compose logs -f
```

5.3 Anbindung an den Reverse Proxy

Damit der Nginx Proxy Manager den Anwendungscontainer erreicht, müssen beide im Netzwerk proxy liegen und der Container unter einem stabilen Namen ansprechbar sein. Dieser Name wird über die Variable PROXY_ALIAS gesetzt und im Proxy Host als Forward Hostname eingetragen. Die mitgelieferte docker-compose.yml bindet das Netzwerk bereits ein; das folgende Beispiel zeigt die maßgeblichen Abschnitte:

```
services:
  wolfisms:
    # ... image, volumes, depends_on ...
    env_file: .env
    restart: unless-stopped
    networks:
      proxy:
        aliases:
          - ${PROXY_ALIAS}    # z. B. wolfisms-app

networks:
  proxy:
    external: true
    name: proxy
```

Kein Port-Mapping für die Anwendung:
Der WolfISMS-Container benötigt keine ports-Angabe. Er wird ausschließlich über das interne Netzwerk proxy angesprochen. Ein zusätzlich veröffentlichter Port würde die Anwendung unter Umgehung der TLS-Terminierung direkt per HTTP erreichbar machen – und damit die Anmeldung wegen der Secure-Cookies zusätzlich unbrauchbar.

Die verbindliche docker-compose.yml ist Bestandteil des Installationspakets. Das obige Beispiel zeigt nur die für die Proxy-Anbindung maßgeblichen Abschnitte und ersetzt die mitgelieferte Datei nicht.

6. Schritt 4 – Proxy Host und Zertifikat einrichten

Die Veröffentlichung erfolgt anschließend in der NPM-Oberfläche unter Hosts → Proxy Hosts → Add Proxy Host.

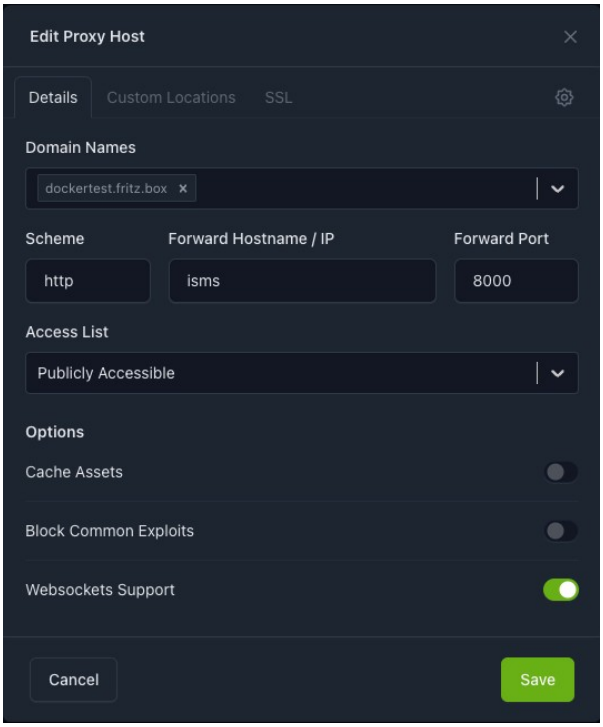
6.1 Reiter „Details“

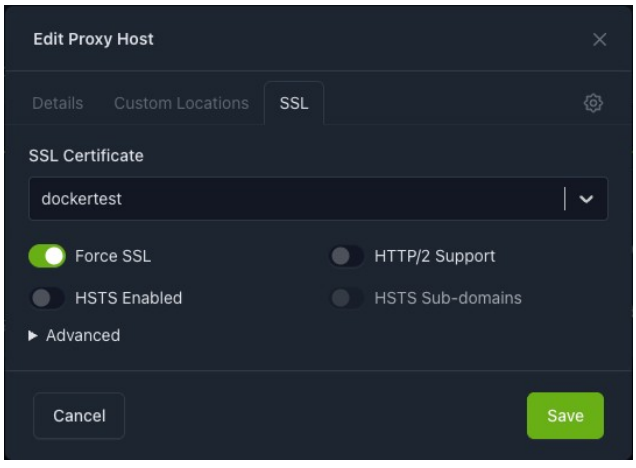
Feld	Wert
Domain Names	isms.ihre-domain.de
Scheme	http
Forward Hostname	wolfisms-app (der in PROXY_ALIAS gesetzte Name)
Forward Port	80
Block Common Exploits	aktiviert
Websockets Support	aktiviert

6.2 Reiter „SSL“

- **SSL Certificate**: „Request a new SSL Certificate“ auswählen
- **Force SSL**: aktivieren – leitet HTTP-Aufrufe automatisch auf HTTPS um
- **HTTP/2 Support**: aktivieren
- **E-Mail-Adresse**: für die Ablaufbenachrichtigungen von Let's Encrypt hinterlegen und die Nutzungsbedingungen bestätigen

Let's Encrypt setzt für die Standard-Challenge voraus, dass der Domainname öffentlich auf die IP-Adresse des Hosts zeigt und Port 80 von außen erreichbar ist. Ist das nicht gegeben – etwa in einer rein internen Umgebung –, muss stattdessen eine DNS-Challenge über den DNS-Anbieter konfiguriert oder ein eigenes Zertifikat hochgeladen werden.





6.3 Installation prüfen

Prüfpunkt	Erwartetes Ergebnis
https://isms.ihre-domain.de aufrufen	Die Loginseite erscheint, das Zertifikat wird als gültig angezeigt
HTTP-Aufruf derselben Adresse	Automatische Weiterleitung auf HTTPS (Force SSL)
Anmeldung	Weiterleitung auf das Dashboard – gelingt die Anmeldung nicht, siehe Kapitel 9
Containerprotokoll	docker compose logs ohne Traceback-Einträge
Adminbereich	Der Menüpunkt „Updates“ fehlt – das ist im Container-Modus so vorgesehen (siehe Kapitel 7)

Unmittelbar nach der ersten Anmeldung:
1) Standardpasswort des Admin-Benutzers ändern
2) Lizenzdatei customer.lic einspielen (Admin → Lizenz)
3) Container neu starten, damit die Lizenz und der daraus abgeleitete Secret Key wirksam werden: docker compose restart

7. Betrieb im Container-Modus

Mit der Umgebungsvariablen WOLFISMS_CONTAINER=1 erkennt die Anwendung, dass sie in einem Container läuft (config.container_mode). Das ändert das Verhalten an zwei Stellen.

7.1 In-App-Updater gesperrt

Der gesamte Update-Bereich einschließlich /admin/updates ist gesperrt und antwortet mit HTTP 404; die Kachel im Adminbereich wird ausgeblendet. Das ist beabsichtigt: Ein Update im laufenden Container würde beim nächsten Neustart des Images verloren gehen.

Aktualisierungen erfolgen im Docker-Betrieb ausschließlich über ein neues Image:

```
docker compose pull
docker compose up -d
```

7.2 Konfiguration ausschließlich aus der Umgebung

Außerhalb von Docker liest config.py die Zugangsdaten aus der Datei /etc/wolfisms/secret.env. Im Container existiert diese Datei in der Regel nicht – das ist kein Fehler: Fehlt sie, verwendet die Anwendung ausschließlich die gesetzten Umgebungsvariablen. Genau diese liefert die Datei .env über docker compose.

Abweichender Pfad:
Soll doch eine Env-Datei im Container verwendet werden, lässt sich ihr Pfad über die Variable WOLFISMS_ENV_FILE setzen. Bereits gesetzte Umgebungsvariablen haben dabei immer Vorrang vor den Werten aus der Datei.

7.3 Datenhaltung und Sicherung

Persistent zu halten sind die Datenbank sowie das Upload-Verzeichnis. Beide müssen als Volume eingebunden sein, sonst gehen sie beim Austausch des Containers verloren.

Daten	Ort	Hinweis
Datenbank	Volume des Datenbankcontainers bzw. externer Datenbankserver	Enthält sämtliche ISMS-Inhalte
Hochgeladene Dateien	UPLOAD_DIR (Standard /var/www/isms/wolfisms/uploads)	Anhänge, Playbooks, Logo, Vorlagen und Briefpapier
Sessions	SESSION_DIR (Standard /var/www/isms/wolfisms/sessions)	Muss beschreibbar sein; ein Verlust meldet lediglich alle Benutzer ab. Die Rechte setzt die Anwendung beim Start selbst auf 0700, die Sitzungsdateien auf 0600
Lizenzdatei	license/customer.lic im Anwendungsverzeichnis	Nach einem Image-Wechsel prüfen, ob sie noch vorhanden ist

Die Sicherung selbst erfolgt wie bei der klassischen Installation über den Adminbereich unter /admin/backup oder über den Direktlink /backup/download. Das Archiv enthält den Datenbank-

Dump und das Upload-Verzeichnis; Anwendungscode, Konfiguration und Lizenzdatei sind nicht enthalten und müssen separat gesichert werden. Einzelheiten enthält das Kapitel „Datensicherung (Backup)" der Administrationsdokumentation.

7.4 Protokolle

Zweck	Befehl
Laufende Ausgabe verfolgen	docker compose logs -f
Nur die Anwendung	docker compose logs -f wolfisms
Letzte 200 Zeilen	docker compose logs --tail=200
Zugriffe und Zertifikate des Proxys	docker compose logs -f npm (im Verzeichnis /opt/npm)

8. Konfigurationsreferenz (.env)

Alle Parameter werden als Umgebungsvariablen gesetzt. Die Auflösung erfolgt in dieser Reihenfolge: bereits gesetzte Umgebungsvariablen, danach die Env-Datei, zuletzt die im Code hinterlegten Vorgabewerte. Für DB_PASSWORD gibt es bewusst keinen Vorgabewert – fehlt die Variable, startet die Anwendung nicht.

8.1 Pflichtangaben

Variable	Beispiel	Bedeutung
DB_PASSWORD	(starkes Passwort)	Datenbankpasswort – einziger Wert ohne Vorgabe
DB_USER	isms	Datenbankbenutzer (Vorgabe: wolf)
DB_HOST	db	Hostname des Datenbankservers; im Compose-Verbund der Servicename
DB_NAME	isms	Name der Datenbank
ISMS_BASE_URL	https://isms.ihre-domain.de	Öffentliche Basis-URL – wird für Links in Erinnerungs-E-Mails benötigt
FLASK_SECRET_KEY	(64 Hex-Zeichen)	Fallback-Session-Schlüssel. Liegt eine gültige Lizenz vor, wird er ignoriert und der Schlüssel aus der Lizenz-ID abgeleitet
WOLFISMS_CONTAINER	1	Aktiviert den Container-Modus (siehe Kapitel 7)
PROXY_ALIAS	wolfisms-app	Netzwerk-Alias des Containers; im Proxy Host als Forward Hostname einzutragen

Einen sicheren Zufallswert für FLASK_SECRET_KEY erzeugen:

```
python3 -c "import secrets; print(secrets.token_hex(32))"
```

8.2 Pfade und Instanz

Variable	Vorgabe	Bedeutung
WEBPATH	(leer)	URL-Pfadpräfix. Beim Betrieb unter einer eigenen Domain leer lassen; nur setzen, wenn die Anwendung unter einem Unterpfad wie /isms läuft
PATHNAME	isms	Verzeichnisname der Instanz
UPLOAD_DIR	/var/www/isms/wolfisms/uploads	Pfad für hochgeladene Dateien – als Volume einbinden
SESSION_DIR	/var/www/isms/wolfisms/sessions	Pfad für serverseitige Sessions – muss beschreibbar sein
WOLFISMS_ENV_FILE	/etc/wolfisms/secret.env	Abweichender Pfad einer Env-Datei im Container

8.3 Sprache

Variable	Vorgabe	Bedeutung
DEUTSCH	True	Deutsche Oberfläche zulassen

Variable	Vorgabe	Bedeutung
ENGLISCH	False	Englische Oberfläche zulassen. Mit DEUTSCH=0 und ENGLISCH=1 läuft die Installation rein englisch

8.4 Single Sign-On (optional)

Variable	Bedeutung
MS_AUTH	1 = Anmeldung über Microsoft Entra ID aktivieren
GOOGLE_AUTH	1 = Anmeldung über Google aktivieren
SSO_ALLOWED_DOMAINS	Kommaseparierte Allowlist zugelassener E-Mail-Domains. Eine leere Liste lässt niemanden zu – das ist der sichere Vorgabewert
SSO_AUTO_CREATE	1 = unbekannte SSO-Benutzer automatisch anlegen (Vorgabe 0)
MS_TENANT_ID / MS_CLIENT_ID / MS_CLIENT_SECRET / MS_REDIRECT_URI	Zugangsdaten und Callback-URL für Microsoft
GOOGLE_CLIENT_ID / GOOGLE_CLIENT_SECRET / GOOGLE_REDIRECT_URI	Zugangsdaten und Callback-URL für Google

Die Callback-URL muss auf den öffentlichen Hostnamen zeigen, den der Reverse Proxy bedient – nicht auf den Containernamen.

8.5 Weitere Variablen

Variable	Bedeutung
UPDATE_PROXY	Proxy für ausgehende Verbindungen. Im Container-Modus ohne Wirkung auf den gesperrten Updater
RATELIMIT_STORAGE_URI	Speicher für die Rate-Limit-Zählung. Die Vorgabe memory:// zählt je Arbeitsprozess – mit mehreren Workern vervielfacht sich damit jedes Limit. Für solche Installationen auf einen gemeinsamen Speicher setzen, etwa redis://redis:6379
ISMS_DEBUG	1 = Flask-Debug-Modus einschalten. Nur zur Fehlersuche und danach wieder entfernen

ISMS_DEBUG darf im Produktivbetrieb nicht gesetzt bleiben. Der Debug-Modus schaltet PROPAGATE_EXCEPTIONS ein, deaktiviert das Template-Caching und ist in jeder Härtungs-Baseline ein Finding.

9. Fehlersuche

Symptom	Ursache und Abhilfe
„network proxy declared as external, but could not be found“	Das Docker-Netzwerk fehlt. Mit docker network create proxy anlegen (Kapitel 3) und die Container erneut starten
NPM meldet „502 Bad Gateway“	Der Anwendungscontainer ist über den eingetragenen Forward Hostname nicht erreichbar. Prüfen, ob der Container läuft, ob er im Netzwerk proxy liegt und ob PROXY_ALIAS mit dem im Proxy Host eingetragenen Namen übereinstimmt
Loginseite erscheint, die Anmeldung schlägt aber wortlos fehl	Der Zugriff erfolgt über HTTP statt HTTPS. Session-Cookies sind Secure=True und werden über unverschlüsselte Verbindungen nicht gesetzt. „Force SSL“ im Proxy Host aktivieren
Container startet nicht, Protokoll nennt die Datenbank	DB_PASSWORD fehlt oder die Datenbank ist noch nicht bereit. Die Variable ist Pflicht und hat bewusst keinen Vorgabewert; die Startreihenfolge über depends_on prüfen
Zertifikatsanforderung schlägt fehl	Der DNS-Eintrag zeigt nicht auf den Host oder Port 80 ist von außen nicht erreichbar. Beides prüfen oder auf eine DNS-Challenge ausweichen
Links im ISMS zeigen auf http:// oder den Containernamen	Der Proxy übergibt X-Forwarded-Proto bzw. X-Forwarded-Host nicht, oder ISMS_BASE_URL ist falsch gesetzt. Beide Angaben prüfen
Der Menüpunkt „Updates“ fehlt im Adminbereich	Kein Fehler: Im Container-Modus ist der In-App-Updater gesperrt. Aktualisierung über ein neues Image (Kapitel 7.1)
Nach dem Einspielen der Lizenz gilt weiterhin der Demo-Modus	Die Lizenzprüfung ist zwischengespeichert. Container neu starten: docker compose restart
Hochgeladene Dateien sind nach einem Update verschwunden	UPLOAD_DIR war nicht als Volume eingebunden. Volume ergänzen und die Dateien aus der letzten Sicherung wiederherstellen

10. Checkliste

✓	Aufgabe
☐	DNS-Eintrag zeigt auf die öffentliche IP-Adresse des Docker-Hosts
☐	Ports 80 und 443 sind von außen erreichbar
☐	Port 81 ist auf das interne Netz beschränkt
☐	Docker-Netzwerk proxy angelegt
☐	Nginx Proxy Manager gestartet, Standard-Zugangsdaten geändert
☐	.env aus .envdefault erstellt und vollständig ausgefüllt
☐	DB_PASSWORD gesetzt und FLASK_SECRET_KEY durch einen Zufallswert ersetzt
☐	WOLFISMS_CONTAINER=1 gesetzt
☐	PROXY_ALIAS gesetzt und im Proxy Host als Forward Hostname eingetragen
☐	ISMS_BASE_URL auf den öffentlichen Hostnamen gesetzt
☐	Volumes für Datenbank und UPLOAD_DIR eingebunden
☐	Proxy Host angelegt, Zertifikat angefordert, Force SSL und HTTP/2 aktiviert
☐	Anmeldung über HTTPS erfolgreich getestet
☐	Standardpasswort des Admin-Benutzers geändert
☐	Lizenzdatei eingespielt und Container neu gestartet
☐	Zweiter Admin-Account als Notfallzugang angelegt
☐	Backup über /admin/backup getestet und extern abgelegt
☐	ISMS_DEBUG nicht gesetzt
☐	.env nicht in ein Git-Repository eingecheckt

11. Weiterführende Dokumentation

Dokument	Inhalt
WolfISMS Admindokumentation	Vollständige Systemadministration: Benutzer- und Rechteverwaltung, Datensicherung, Lizenzverwaltung, REST-API, Zwei-Faktor-Authentifizierung, Single Sign-On, KI-Einstellungen und Sicherheitshinweise. Dort steht auch, was im Containerbetrieb anders läuft als bei der klassischen Installation auf Apache mit mod_wsgi
WolfISMS Benutzerdokumentation	Beschreibung aller Fachmodule für Anwender
WolfISMS REST-API	Schnittstellenbeschreibung für die Anbindung externer Systeme: Incidents, Maßnahmen, Assets, Risiken und Lieferanten – mit allen Feldern, Filtern, Wertelisten und einem vollständigen Beispiel-Sync-Client

WolfISMS – Wolf's Informations-Sicherheits-Management-System

Update-Server: https://update.wolfisms.de
Version: 3.63 | Stand: August 2026